TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: M 2612 – Elektrotechnika a informatika
Studijní obor: 3902T005 – Automatické řízení a inženýrská informatika

Vizualizace datových komunikací

Visualization of data communication

Diplomová práce

Autor: **Vít Bartůněk**
Vedoucí práce: **RNDr. Pavel Satrapa, Ph.D.**

V Liberci 2007

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Katedra aplikované informatiky Akademický rok: 2006/2007

ZADÁNÍ DIPLOMOVÉ PRÁCE

Jméno a příjmení: Vít Bartůněk

studijní program: M 2612 – Elektrotechnika a informatika

obor: 3902T005 – Automatické řízení a inženýrská informatika

Vedoucí katedry Vám ve smyslu zákona o vysokých školách č.111/1998 Sb. určuje tuto diplomovou práci:

Název tématu: **Vizualizace datových komunikací**

Zásady pro vypracování:

1. Využijte a rozšiřte své znalosti problematiky statistického vyhodnocení datových komunikací získané při řešení ročníkového projektu Analýza a řízení datových přenosů Ethernet sítí v prostředí Linux.
2. Zvyšte výkon jádra aplikace pro zobrazování datových toků a zrychlete její odezvy.
3. Rozšiřte schopnosti aplikace o možnost cíleného zobrazení datových toků vybraných počítačů na základě uživatelova zadání.
4. Implementujte rozpoznávání mimořádných a kritických situací podle uživatelem specifikovaných parametrů.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce.

Datum

Podpis

Poděkování

Na tomto místě bych rád poděkoval vedoucímu diplomové práce RNDr. Pavlovi Satrapovi, Ph.D. za podněty, připomínky a cenné rady, které mi poskytl. V neposlední řadě bych také rád poděkoval celé mojí rodině za finanční a morální podporu po celou dobu studia na TUL.

Anotace:

Práce se zabývá monitorováním a optimalizací datových komunikací ethernet sítí v prostředí operačního systému Linux. Obě tyto části jsou řešeny samostatně, ve výsledku však spolu vytvářejí QoS systém, který se snaží zajistit kvalitu služby pro všechny stanice ve vnitřní síti. Celé řešení je realizováno formou více nezávislých aplikací, které spolu komunikují pomocí architektury klient-server. Koncepce rozdělení celkového problému na více částí umožňuje celkově efektivnější řešení a úpravu jednotlivých částí samostatně.

Systém pro monitorování provozu umožňuje sledovat využití sítě na specifických službách pro každou stanici vnitřní sítě a zobrazovat cílené statistiky na základě definovaných parametrů.

Na základě algoritmů pro řízení a zdrojových dat získaných při monitorování je implementován snadno konfigurovatelný QoS systém s podporou „Fair User Policy“.

Abstract:

This Diploma thesis is focused on monitoring and optimalization of data communication in ethernet networks at Linux Operating system. These both parts are solved seperately, however in the final effect they create together the QoS system that provide quality of the service for all computer stations in a local area network. The resulting solution is realized as a system of several independent applications based on the client server communication model. The intention to split that entire task into several segments makes possible more effective solution as a final effect and provides free developing of each part separately.

Monitoring system has a capability to scan an utilization of a network regarding specific services for each computer separately and to display targeted statistics by means of defined parameters.

The easy configurable QoS system with „Fair User Policy“ support is implemented on the basis of data from monitoring and controlling algorithms.

Obsah

Přílohy

Seznam použitých zkratek

AJAX	-	Asynchronous JavaScript and XML
CRON	-	plánovací služba pro operační systémy Unix
DNAT	-	Destination NAT, změna cílové IP adresy
Egress	-	odchozí provoz na síťovém rozhraní
HTTP	-	HyperText Transfer Protocol, protokol pro www službu
FTP	-	File Transfer Protocol, protokol pro přenos souborů v síti
FUP	-	Fair User Policy, omezování konektivity v závislosti na objemu stažených dat
SSL	-	Secure Sockets Layer, protokol zabezpečený šifrováním
IANA	-	autorita přiřazující IP adresy a další číselné identifikátory v Internetu
Ingress	-	příchozí provoz na síťovém rozhraní
IP	-	Internet Protocol, pomocí kterého komunikují zařízení v Internetu
Iptables	-	síťový nástroj sloužící jako firewall pro operační systémy Linux
ITU-T	-	telekomunikační sektor mezinárodní telekomunikační unie
L7filter	-	klasifikátor využívající informace z aplikačního protokolu
L7 protocols	-	vzory protokolů pro Layer7 filter
MAC	-	podvrstva linkové vrstvy, mimo jiné definuje také hardwarové adresy
NAT	-	Network Address Translation, změna IP adres směrovačem
NULL	-	ukazatel nikam
OT	-	nerozpoznaný, jiný typ služby
P2P	-	„peer to peer“, typ komunikace v síti využívaný pro sdílení souborů
PERL	-	programovací jazyk
PHP	-	skriptovací programovací jazyk
PNG	-	Portable Network Graphics, bezeztrátový obrazový formát dat
Policing	-	tvarovač příchozího provozu na síťovém rozhraní
QoS	-	Quality of Service, kvalita služby
Session	-	paměť pro proměnné sloužící k ukládání dat na webovém serveru
Shaper	-	tvarovač odchozího provozu na síťovém rozhraní
SNAT	-	Source NAT, změna zdrojové IP adresy
SIGCHLD	-	systémový signál vyvolaný při ukončení procesu
TCP/IP	-	skupina protokolů pro komunikaci v síti
ToS	-	Type of Service, typ služby přenášené IP protokolem
VNC	-	Virtual Network Computing, vzdálené připojení k počítači
VOIP	-	přenos digitalizovaného hlasu IP protokolem

1. Úvod

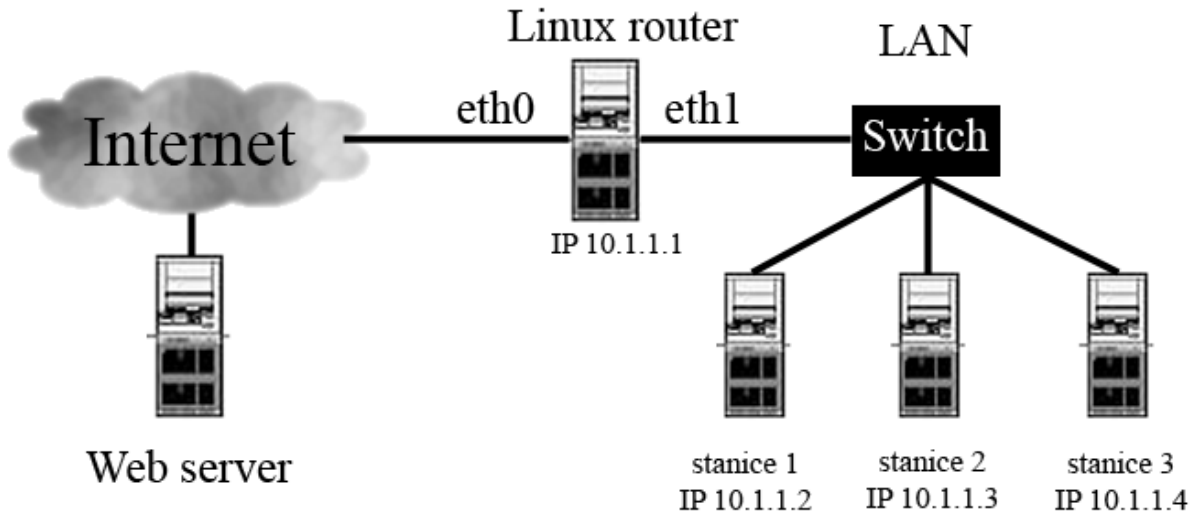
Cílem této diplomové práce je vizualizovat a řídit datovou komunikaci stanic v lokální síti, které využívají služby Internetu. Data pro vizualizaci jsou získána na směrovači, který spojuje lokální síť (LAN) a vnější síť (Internet). Procházející data mezi vnitřní a vnější sítí je nutno analyzovat a rozlišit datové toky jednotlivých stanic. V rámci datového toku každé stanice dále rozlišujeme jednotlivé aplikační protokoly (služby).

Za účelem rozlišení procházejících dat směrovačem na jednotlivé služby je využito průchodu celkového datového toku nástrojem „Iptables“. Iptables nám pak umožňuje získávat informace o množství dat (zvlášť pro jednotlivé aplikační protokoly), která projdou směrovačem.

V závěru práce je na základě statistik na směrovači implementován QoS systém, který ovlivňuje chování datových toků pro jednotlivé stanice.

1.1 Model realizace

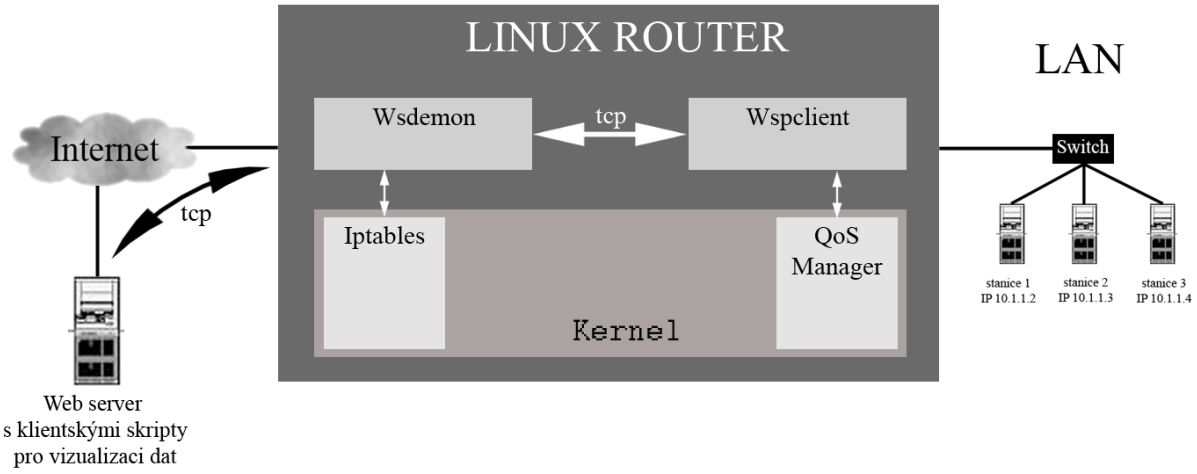
Pro realizaci diplomové práce bylo využito domácí lokální sítě. Síť je složená z několika stanic ve vnitřní síti, která je připojena pomocí směrovače k Internetu. Model této sítě je zobrazen na Obr. 1 - Model testovací sítě.



Obr. 1 - Model testovací sítě

Veškeré datové toky mezi vnitřní sítí a Internetem procházejí přes počítač označený „Linux router“.

Koncepce diplomové práce spočívá v řešení jednotlivých cílů odděleně formou samostatných aplikací. Jednotlivé aplikace spolu komunikují po síti pomocí architektury klient server. Nastíněné řešení je zobrazeno na Obr. 2 - Model realizace. „Linux router“ je počítač s operačním systémem Linux, sloužící jako směrovač k propojení vnitřní sítě s Internetem. Na tomto počítači je spuštěna aplikace „Wsdemon“ a „Wspclient“.



Obr. 2 - Model realizace

Wsdemon tvoří jádro celého systému a realizuje pomyslnou vrstvu, která slouží klientským aplikacím pro přístup k informacím. Aplikace je vytvořena v programovacím jazyku C a využívá zdrojové kódy Iptables. Poskytuje informace o procházejících datech tímto počítačem, záznamy o stanicích v síti, parametry připojení jednotlivých stanic do sítě a obsahuje algoritmy pro vypočítání parametrů využívaných FUP systémem. Pro přístup k datům je spuštěna jako souběžný server komunikující na TCP protokolu.

Wspclient je klientská aplikace-skript vytvořený v jazyce PERL. Na základě zadaných parametrů získaných z aplikace Wsdemon aplikuje parametry QoS do jádra systému směrovače. QoS pak ovlivňuje průchodnost sítě k jednotlivým koncovým stanicím.

Pro vizualizaci dat jsou vytvořeny skripty, které generují grafy zobrazované na webovém serveru. Pro generování grafů dlouhodobých a aktuálních „realtime“ statistik jsou použity odlišné technologie a aplikační skripty, které jsou na sobě nezávislé. Pro oba typy klientských skriptů jsou data získávány po síti z aplikace Wsdemon. Grafy zobrazující informace o přenášených datech v dlouhodobém horizontu jsou generovány pomocí skriptu Wspclient a nástroje RRDTool, který obsahuje vlastní databázový systém. Pro zobrazování aktuálně procházejících dat směrovačem byly vytvořeny PHP skripty, které využívají rozšiřujících modulů „PEAR“.

V případě je-li směrovač uzpůsobený zároveň jako webový server, mohou být vizualizační skripty spouštěny přímo na směrovači. Zobrazování realtime statistik však klade nemalé nároky na systémové zdroje a proto je lepší tyto grafy generovat na jiném webovém serveru.

1.2 Navázání na projekt

Tato diplomová práce navazuje na ročníkový projekt „Analýza a řízení datových přenosů ethernet sítí v prostředí LINUX“. Na základě předchozí zkušenosti s touto problematikou byly stanoveny cíle tak, aby se rozšířily možnosti statistického vyhodnocení datových toků a odstranily problémy, které se při realizaci projektu objevily. Rád bych se zde zmínil o odlišnostech současné a předchozí realizace v jednotlivých bodech:

1)Realizace přístupu k datům

2)Klient-Server přenos dat

Původní řešení v ročníkovém projektu:

Ad 1) Přístup k datům byl řešen pomocí skriptu v jazyce PERL, který používal k nastavení struktury Iptables. Zpětné získávání informací se provádělo spuštěním programu s požadovanými parametry. Výstup dat z Iptables, který byl posílán na konzoli, byl brán jako vstup zdrojových informací pro skript. Tyto informace obsahovaly jak užitečná tak neužitečná data a proto bylo nutné informace rozparsovat na jednotlivé části. Toto řešení se ukázalo pro rozsáhlejší struktury Iptables a potřebu rychlého čtení informací o přenášených datech nedostatečné.

Ad 2) Řešení přenosu dat pomocí architektury klient server vypracoval můj kolega, se kterým jsem na ročníkovém projektu spolupracoval. Bohužel server nepodporoval současné připojení více klientů zároveň a docházelo k neočekávanému ukončení serverové aplikace.

Nové řešení použité v diplomové práci:

ad 1) Jedinou možností jak dosáhnout rychlejších odezev čtení informací z Iptables bylo využití knihoven libiptc. Jsou určeny pro jazyk C a umožňují přímý přístup ke strukturám Iptables v paměti jádra. Použití těchto knihoven je výhodné pouze pro čtení dat. Vkládání pravidel do řetězců pomocí těchto knihoven je zbytečně komplikované a neexistuje pro ně žádná dokumentace. Rozhodl jsem se tedy využít přímo hlavičkové a zdrojové soubory Iptables, pomocí kterých dochází k veškerému zápisu do paměti těchto struktur. Tento přístup zajistil dostatečné zvýšení výkonnosti a umožnil použít řešení jak pro malé, tak i velmi rozsáhlé a komplikované sítě. Porovnání výkonnosti jednotlivých operací je zobrazeno v sekci 6 Porovnání výkonnosti.

Ad 2) Rozšíření předchozího řešení se mi zdálo neefektivní a proto jsem naprogramoval novou serverovou část i jednotlivé klientské aplikace, které se statistikami dále pracují. Server je naprogramován tak, aby umožnil současné obsloužení více klientů zároveň. Klienti sloužící pro vizualizaci statistik jsou vytvořeni formou webové aplikace. Graficky zpracované statistiky jsou přístupné na webovém serveru bez nutnosti použití externí klientské aplikace.

2. Principy komunikace v Internetu

2.1 Internet a TCP/IP

Zkratka TCP/IP se používá k označení skupiny komunikačních protokolů využívaných v Internetu. Komunikační protokol je množina pravidel, které určují syntaxi a význam jednotlivých zpráv při komunikaci.

Síťová komunikace protokolu TCP/IP je rozdělena do čtyř vrstev (aplikační, transportní, síťové a vrstvy síťového rozhraní), které vyjadřují hierarchii činností. Výměna informací mezi vrstvami je přesně definována. Každá vrstva využívá služeb vrstvy nižší a naopak poskytuje své služby vrstvě vyšší. Hlavním důvodem rozdělení protokolu do vrstev je zajištění kompatibility při komunikaci různých systémů nebo komunikaci v nehomogenních sítích.

Mezi základní protokoly, které TCP/IP zahrnuje patří:

IP Internet Protocol je základní protokol síťové vrstvy a celého Internetu. Vyšším vrstvám poskytuje síťovou službu, která je nespojovaná a nezaručuje ani doručení dat. Pokud aplikace vyžaduje spojované či spolehlivé spojení, je ho nutné toto implementovat ve vyšší vrstvě. IP protokol leží na úrovni síťové vrstvy ISO/OSI modelu a ke komunikaci používá datagramy. Protokol zajišťuje rozdělení dat na jednotlivé datagramy, které pak v síti putují nezávisle na sobě. Protokol IP je v současnosti ve variantě IPv4 a IPv6, zásadní rozdíl je hlavně ve zvětšení adresového prostoru u IPv6 a vylepšení bezpečnosti i funkcí QoS pro zajištění kvality služeb.

ICMP Internet Control Message Protocol slouží k přenosu řídících hlášení, která se týkají chybových stavů a zvláštních okolností při přenosu. Využívá se například v programu ping pro testování dostupnosti počítače nebo programem traceroute pro sledování cesty paketů k jinému uzlu.

TCP Transmission Control Protocol vytváří spolehlivé a spojované spojení mezi koncovými aplikacemi. Leží na úrovni transportní vrstvy ISO/OSI modelu. Data jsou doručována adresátovi beze ztrát v pořadí v jakém byla vyslána. Spojení je tvořenou „rourou“, na obou koncích roura vyúsťuje soketem, pomocí kterého posíláme a získáváme data z roury. Aplikace mohou vysílat a přijímat zároveň. Nejprve dojde k navázání spojení synchronizačními pakety, poté k samotnému přenosu dat a na závěr se spojení ukončí.

UDP User Datagram Protocol poskytuje přenosovou službu pro aplikace, které nevyžadují spolehlivost. Přenáší datagramy mezi počítači v síti, ale na rozdíl od TCP nezaručuje zda se přenášený paket neztratí, nezmění se pořadí paketů nebo zda některý paket nedoručí vícekrát. Výhodou tohoto protokolu je, že nedochází k navazování spojení, ale posílají se vždy přímo data. Díky tomu je UDP protokol rychlejší a efektivnější. Podobně jako TCP používá čísla portů pro identifikaci aplikačních protokolů (služeb).

2.2 Soket

Pro komunikaci mezi aplikacemi je potřeba definovat mechanizmus, který umožní komunikaci aplikací v Internetu na aplikační vrstvě. Právě pro tento účel slouží soket. Soket obsahuje informace:

- protokol (TCP, UDP, IP)
- IP adresa lokálního síťového rozhraní
- lokální port
- IP adresa vzdáleného síťového rozhraní
- vzdálený port

Pomocí těchto informací je definováno kompletní spojení mezi aplikacemi. Při komunikaci dojde v aplikaci nejprve k vytvoření soketu s danými parametry, poté ke spojení soketu s protější stranou, následuje vlastní výměna dat pomocí tohoto soketu a uzavření soketu.

2.3 NAT (Network Address Translation)

Dříve definovaná adresace internetu odpovídající protokolu IPv4 se ukázala jako nedostatečná z hlediska počtu IP adres. V důsledku pak na všechny počítače v Internetu nezbyla unikátní IP adresa. K vyřešení tohoto problému se dnes používá NAT, který je využit i v navržené modelové síti.

Network Address Translation (NAT) je funkce síťového směrovače a slouží ke změnám údajů v hlavičkách paketů. NAT musí zajistit u odesílaných paketů změnu IP adresy odesílatele (DNAT) a u přicházejících paketů adresu příjemce (SNAT).

Nejčastější použití NAT je takzvaná „maškaráda“. Například máme od poskytovatele připojení k dispozici jednu veřejnou IP adresu, ale potřebujeme k síti připojit větší množství počítačů. Proto veřejnou adresu přidělíme směrovači a ostatním počítačům přidělíme neveřejné (privátní) IP adresy, které jsou k tomuto účelu rezervovány podle RFC.

Když počítač z lokální sítě odesílá paket do vnější sítě (např. Internetu), odešle jej se svou zdrojovou IP adresou a portem. Při průchodu směrovačem jsou v paketech přepsány zdrojové IP adresy na veřejnou IP adresu směrovače a číslo zdrojového portu na port, který směrovač přidělí. Směrovač si zároveň toto přidělení uloží do své převodní tabulky (obsahuje veškeré informace o vzájemném mapování jednotlivých adres a portů).

NAT lze využít i v případě když máme ve vnitřní síti umístěný server, který nemá veřejnou IP adresu a rádi bychom na něho přesměrovali požadavky.

3 Získání informací pro vizualizaci

K získání informací o datech protékajících směrovačem je využito nástroje Iptables. Iptables je součástí jádra systému a veškeré pakety, které projdou přes směrovač, jsou zpracovány pomocí tohoto nástroje. V této kapitole je rozebrán princip nastavení struktury Iptables tak, abychom mohli získávat potřebná data.

3.1 Úvod do Iptables

Iptables je mocný nástroj, který umožňuje linuxovému nebo unixovému systému plně pracovat se síťovou komunikací a můžeme si pomocí něho snadno sestavit různé druhy firewallů (stavové či transparentní) s podporou adres IPv4 i IPv6 nebo vytvořit směrovač pro lokální síť s překládáním adres (NAT). Pro tento nástroj existuje velká řada rozšiřujících modulů pro klasifikaci paketů. Mezi ty zajímavé můžeme například zařadit klasifikaci podle délky paketu, času, TTL, obsahu nebo s fuzzy logikou. Veškeré možnosti a moduly, které Iptables nabízí jsou na stránkách skupiny Netfilter [3]. V současnosti je Iptables součástí snad každé distribuce a proto se instalací tohoto nástroje není třeba zabývat.

3.1.1 Vysvětlení firewallu a nastavení Iptables jako směrovače

Vycházíme z modelové situace, kdy máme linuxový směrovač se dvěmi síťovými kartami, přičemž na jedné straně rozhraní (eth0) je Internet a na druhé straně (eth1) lokální síť. Každý IP datagram s sebou nese mimo vlastních užitečných dat také hlavičku obsahující zejména IP adresu původce i adresáta, zdrojový a cílový port. Paketový firewall je pak jakýmsi filtrem, který na základě informací z hlavičky paketu či jeho obsahu rozhoduje o tom, které pakety mohou být připuštěny až na aplikační vrstvu ISO/OSI modelu, tedy ke konkrétním aplikacím nebo které naopak smějí opustit počítač.

Každý paket, ať už je jeho původ jakýkoliv, prochází systémem řetězců, které tvoří filtrovací tabulku. Pro každý řetězec jsou definována pravidla, která určují zda-li má paket odbočit do daného řetězce, či pokračovat dále v cestě tabulkou až na její konec. Každé takové pravidlo konkrétního řetězce obsahuje vlastní počítadlo paketů a bytů, které obsahuje informaci kolik paketů bylo pomocí tohoto pravidla propuštěno do daného řetězce. Řetězce a pravidla pak lze přidávat na výstupy dalších řetězců.

Tímto způsobem lze vytvářet přesně definované struktury, pomocí kterých můžeme pakety nejen efektivně filtrovat, ale zároveň také získáváme informaci o tom jaké množství dat jednotlivými řetězci prošlo. Jakým způsobem je s paketem na konci své pouti řetězci naloženo definuje výstupní hodnota řetězce, do kterého paket vstoupil a nabývá hodnot:

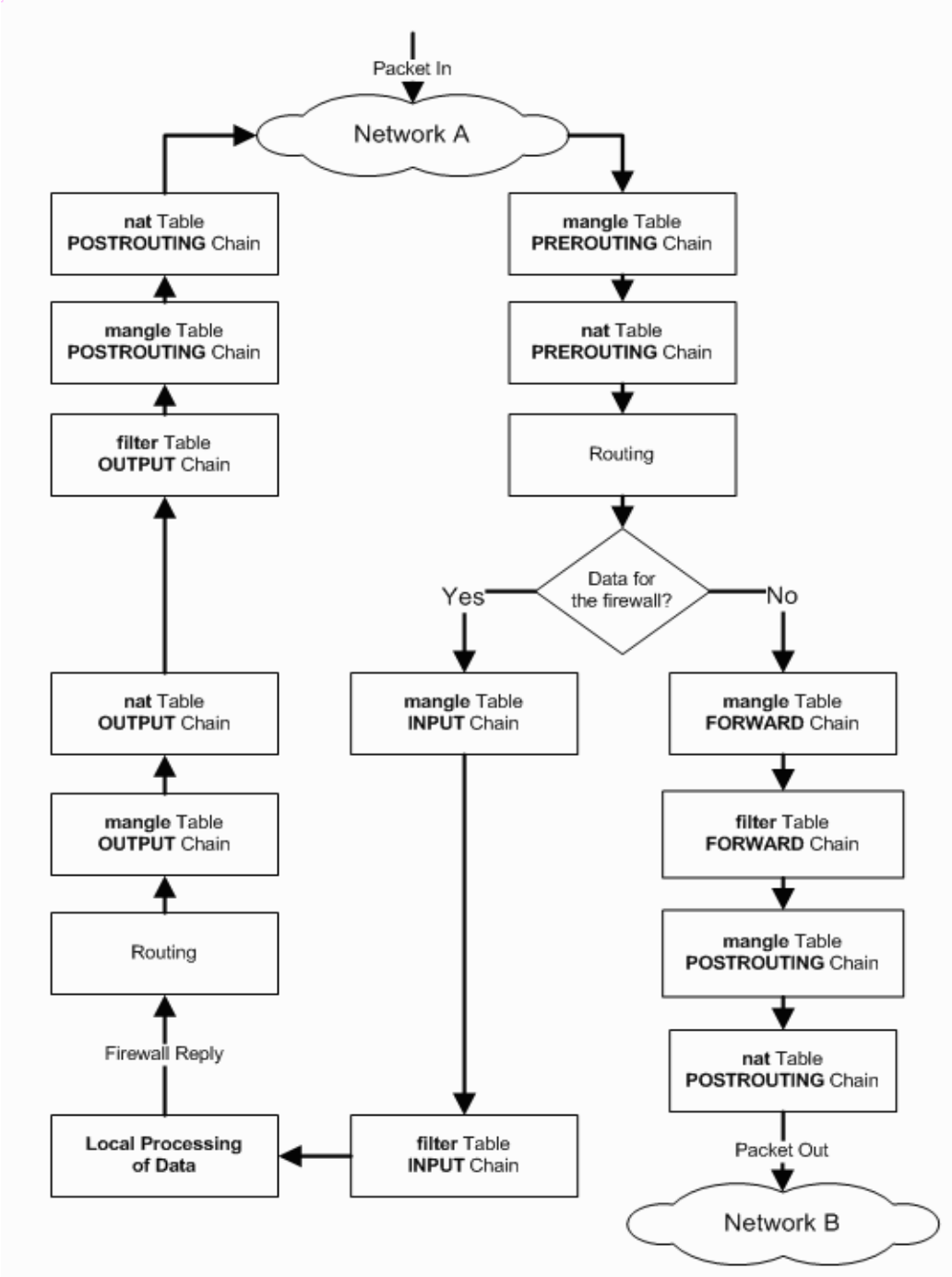
- ACCEPT - paket je propuštěn dále
- REJECT - paket je vrácen zpět odesílateli
- DROP - paket je zahozen
- LOG - záhlaví paketu je zapsáno do systémového logu

V případě, že na konci řetězce není nastavena výstupní hodnota, paket putuje dále tabulkou řetězců v pořadí jakým byly jednotlivé řetězce a pravidla pro směrování do tabulky vloženy. Standardně Iptables definuje řetězce PREROUTING, INPUT, FORWARD, OUTPUT, POSTROUTING, do kterých jsou pakety směřovány automaticky podle obsahu IP hlavičky.

Paketový filtr si můžeme představit jako potrubí a jednotlivé řetězce jako ventily propouštějící pouze vybrané pakety. Na začátku se snaží jádro rozhodnout, zda-li je příchozí paket určen pro tento počítač nebo je potřeba jej přesměrovat jinam.

- Je-li adresátem on sám, předá paket k dalšímu zpracování do vstupního (INPUT) řetězce. Pokud vyhoví stanoveným filtrovacím pravidlům, dostane jej ke zpracování některý z lokálních programů poslouchající na cílovém portu.
- Pokud je datagram určen někomu jinému a počítač je zkonfigurován jako směrovač (tedy pokud je povoleno routování paketů proměnnou /proc/sys/net/ipv4/ip_forward nastavenou na 1), paket bude poslán do řetězce FORWARD. Pokud je směrování paketů zakázáno (implicitní stav), bude paket zahozen.
- Poslední možností je, že datagram vytvořil některý z lokálních programů. Potom je směrován do řetězce OUTPUT.

Kromě zmíněných řetězců INPUT, OUTPUT a FORWARD implicitně existují ještě další dva a to PREROUTING a POSTROUTING. Ty se používají především k jiným účelům než k filtrování. Jak je patrné z názvů, PREROUTING je aktivní v době před routováním. Procházejí jím jak pakety určené pro lokální počítač, tak i ty které budou směrovány jinam. Podobně POSTROUTINGem protékají pakety odcházející z našeho počítače, stejně jako směrované datagramy. Tyto řetězce mají zvláštní význam při překladu adres (NAT). Průchod jádrem Iptables je zřejmý z Obr. 3 - Průchod paketů strukturou Iptables.



Obr. 3 - Průchod paketů strukturou Iptables [10]

3.1.2 Syntaxe Iptables

Program se spouští s několika parametry. Prvním je místo určení, kam chceme pravidlo zařadit. Chceme-li nějaké pravidlo přidat do řetězce INPUT, lze to provést takto:

„iptables -A INPUT pravidlo“

pravidlo je určeno specifikací, se kterou se zkoumaný paket porovnává.

Například zápisu:

„iptables -A INPUT -p tcp -i eth1 -s 10.1.1.2 --dport 80 -j ACCEPT“

vyhovují všechny IP datagramy TCP protokolu, které byly přijaty síťovou kartou eth1, IP adresa odesílatele je stanice 10.1.1.2 a jsou určeny pro libovolného příjemce na portu 80. Všimněme si, že pokud některý parametr explicitně neuvedeme, bude pravidlo vyhovovat libovolnému parametru z množiny možných. Pokud tedy neurčíme cílovou IP adresu pomocí *"-d"*, výchozí hodnotou bude 0/0, tedy libovolná IP adresa s libovolnou maskou podsítě. Parametrem *"-j"* definujeme výstupní hodnotu řetězce, kterým určujeme co má jádro vykonat, pokud zkoumaný paket danému pravidlu vyhovuje. V našem příkladu bude paket propuštěn dále ke zpracování.

Syntaxe Iptables může být velice různorodá a způsobů jak specifikovat množinu datagramů může být mnoho. Vyčerpávající popis parametrů Iptables lze nalézt kromě manuálové stránky Iptables také na stránkách skupiny Netfilter [3].

3.2 Realizace nastavení struktury Iptables

K získání podrobných informací o procházejících datech je nutné navrhnout strukturu Iptables tak, aby docházelo k rozdělení celkového toku paketů na datové toky jednotlivých stanic a služeb. Představme si, že chceme sledovat veškerý datový provoz všech stanic vnitřní sítě směrem z internetu pro webovou službu. Pro naší síť zobrazenou na „Obr. 1 - Model testovací sítě“ by to znamenalo rozlišit na linuxovém routeru pakety procházející z internetu k jednotlivým stanicím pomocí zdrojového portu a cílové adresy získané z IP hlavičky každého z paketů. Vložili bychom tři řetězce do tabulky v Iptables nazvané například ST1_DOWN_WWW, ST2_DOWN_WWW, ST3_DOWN_WWW a těmto třem řetězcům nastavili pravidla. Pravidla pak určují zda má paket procházející tabulkou do takového řetězce vstoupit či nikoliv. Vytvoření struktury pro jednu stanici pomocí spouštění Iptables z příkazové řádky by vypadalo takto:

„iptables -N ST1_DOWN_WWW“

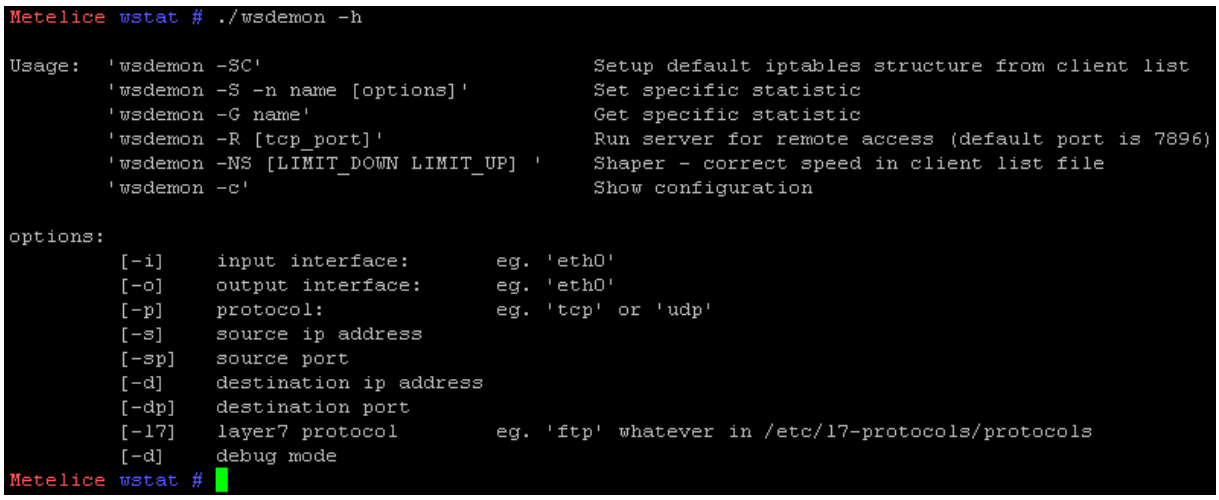
„iptables -A FORWARD -i eth0 -o eth1 -p tcp -s 0/0 –sport 80 -d 10.1.1.2 -j ST1_DOWN_WWW“

Kdybychom pak chtěli definovat strukturu Iptables tak, abychom měli přehled o datové komunikaci v obou směrech a pro více služeb, dostaneme se do situace, kdy ruční nastavení by bylo nemyslitelné z důvodu velkého počtu vložených řetězců a pravidel.

V případě automatického nastavení pro všechny stanice pomocí skriptu vytvořeného například v jazyce PERL zjistíme, že pro cca dvacet stanic v síti trvá nastavení řádově desítky sekund a s počtem stanic lineárně roste. Tento způsob nastavování byl použit při předchozím řešení ročníkového projektu a je pro rozsáhlé sítě nevyhovující.

Nové řešení nastavuje struktury přímo pomocí vnitřní funkce Iptables *„do_command“*. Výsledného zrychlení bylo dosaženo zejména tím, že dojde nejprve k nastavení celé struktury najednou a až následně se zapíší všechny změny do jádra systému.

K nastavení jedné cílené statistiky pak spustíme vytvořenou aplikaci „W*sdemon“* s parametrem *„-S“* a pomocí dalších spouštěcích parametrů tuto statistiku konkretizujeme na cílenou skupinu procházejících paketů. Kompletní výpis možností získáme spuštěním aplikace s parametrem „-h“ jak je vidět na Obr. 4 - Wsdemon nápověda.



Obr. 4 - Wsdemon nápověda

Pokud bychom tedy chtěli získávat informace o množství dat, které nám projde např. ze serverů společnosti Google ke všem stanicím z vnitřní sítě, definovali bychom vstup aplikace takto:

„wsdemon -S -n DOWN_ALL_GOOGLE -s 66.249.93.0/24“

Kde 66.249.93.0/24 definuje rozsah IP adres společnosti Google, popřípadě je možné použít místo IP adresy s maskou přímo URL/maska „www.google.com/24“.

Informace o množství dat, která prošla naší statistikou bychom získali spuštěním programu s parametry „-G DOWN_ALL_GOOGLE“. Tímto bychom si ale úkol nezjednodušili, neboť totéž lze provést přímo spuštěním Iptables a obdrželi bychom stejné výsledky pouze jinou formou výpisu informací. Jak si ale později ukážeme naše aplikace tyto informace může zpřístupnit vzdáleně pomocí síťové architektury klient-server.

Aplikace Wsdemon standardně podporuje nastavení statistik pro celou vnitřní síť, když chceme například sledovat služby HTTP, FTP, Mail, P2P, SSL, Remote, Audio, Video k jednotlivým stanicím v síti pro oba směry, tedy download ve směru z internetu ke stanici a upload ve směru opačném. Předpokládáme, že máme informace o jednotlivých stanicím v síti (tabulku stanic) uložené v souboru, který by mohl vypadat takto:

```
stanice1  10.1.1.2  00:A0:C0:76:2A:F2  2048  512  256  256  2  directconnect  10205  A
stanice2  10.1.1.2  00:A0:C0:76:2A:F2  2048  512  256  256  1  none  none  A
stanice1  10.1.1.2  00:A0:C0:76:2A:F2  1024  256  128  128  1  none  none  A
```

Na jednotlivých řádcích jsou tabulátorem oddělené údaje o stanicích v pořadí jméno, IP, MAC adresa, maximální rychlost download, upload, garantovaná rychlost download, upload, priorita v síti, zakázané protokoly oddělené čárkou, přesměrované porty ke stanici a status. Jelikož Iptables slouží hlavně jako firewall, naše aplikace této skutečnosti využívá a při nastavování jednotlivých řetězců a pravidel se zároveň v aplikaci Wsdemon nastavuje toto:

- firewall na směrovači tak, aby měly přístup do sítě pouze stanice se správnou kombinací IP a MAC adresy.
- Uvedené porty v tabulce se přesměrují ke stanici. Cílová stanice pak může sloužit na tomto portu jako server ve vnitřní síti.
- Podle statusu, který je uveden jako poslední sloupec v tabulce, stanici zpřístupní používání vnější sítě (odpovídá písmenu *„A“*) nebo zakáže písmenem *„N“*.

K automatickému nastavení struktury Iptables podle informací uložených v souboru spustíme aplikaci „w*sdemon -SC“*. Cesta k souboru (s tabulkou stanic) s informacemi o stanicích je uložená v konfiguračním souboru „wstat.config“ aplikace Wsdemon. Podrobnější informace k nastavení jsou v sekci 7.2 Konfigurace aplikace Wsdemon. Další informace uvedené v souboru (s tabulkou stanic) jsou využity v aplikacích, které se na dané informace dotazují po síti. Využívají se například k nastavení propustnosti sítě na směrovači k jednotlivým stanicím nebo k blokování protokolů uvedených ve sloupci zakázané protokoly.

3.3 Klasifikace služeb

K definici pravidel pro jednotlivé řetězce v tabulce, která určují zda-li má daný paket vstoupit do řetězce či nikoliv, standardně využíváme zdrojové a cílové porty, které definují o jakou službu se jedná. Jsou však situace, kdy služba nemá pevně přidělené zdrojové a cílové porty a proto musíme pro nasměrování paketů do řetězců použít metody založené na porovnávání obsahu paketu. Klasifikátory, které využívají ke své funkci data obsažená v paketu, je možné přikompilovat do jádra systému jako rozšíření balíčku Iptables. Mezi nejrozšířenější a nejpropracovanější patří *„L7-filter“*. Principem této klasifikace je porovnávání obsahu paketu se vzorem, který je ve formě textového řetězce obsahem charakterizujícím komunikaci vybraného protokolu. Jednotlivé protokoly *„L7-protocols“* jsou zakomponovány v balíčku l7-filtru a standardně nainstalované v adresáři „/etc/l7-protocols“. Informace o podporovaných protokolech, rychlostech porovnávání, přesnosti klasifikace a další užitečná data může zájemce nalézt na stránkách věnovaných L7-filter [5].

Jelikož klasifikace paketů podle údajů z IP hlavičky paketu je z hlediska výpočetní náročnosti mnohem úspornější než porovnávání obsahu paketu se vzorem, snažíme se definovat strukturu Iptables tak, aby docházelo k minimálnímu počtu takového porovnávání a šetřili čas procesoru.

Ke klasifikaci služeb jsem použil následující pravidla:

Protokol	popis služby	tcp porty	udp porty	l7 protokoly
HTTP	www	80		
FTP	ftp přenos dat			ftp
Mail	elektronická pošta	110, 143, 993, 995	143, 220, 993	
P2P	peer to peer sítě			bittorent, dirrectconnect, edonkey, gnutella, napster
SSL	šifrované spojení	443	ssl	
Remote	vzdálené připojení	22, 23		vnc
Audio	audio streaming, VOIP			httpaudio, shoutcast, skypetoskype
Video	video streaming	8854	8854	httpvideo, quicktime

Pokud je služba klasifikována více parametry, je použit pro klasifikaci nejprve port a poté L7filtr. Filtry pro protokoly, které tvoří pouze podskupinu jiného protokolu je nutné přidat do tabulky jako první. Příkladem jsou protokoly httpaudio a httpvideo, které jsou podskupinou http protokolu.

Použité porty jsou získány ze stránek organizace IANA [6], která je autoritou pro přidělování IP adres, čísel portů a dalších identifikátorů pro služby v síti Internet.

Podívejme se, jak by vypadalo nastavení řetězců a směrovacích pravidel pro jednu stanici ve vnitřní síti, konkrétně pro STANICI1 z naší modelové sestavy. Uvedený postup je popsán pouze principielně, zdrojový kód je pak k dispozici na CD s přílohami.

1. Pro stanici vytvoříme řetězce STANICE1_DOWN, STANICE1_UP a řetězce reprezentující jednotlivé služby pro oba směry, například pro FTP tedy STANICE1_DOWN_FTP, STANICE1_UP_FTP.

2. Nasměrujeme pakety procházející z vnější sítě do vnitřní do řetězce STANICE1_DOWN podle hlavičky cílové IP adresy a v opačném směru pak podle zdrojové IP adresy do řetězce STANICE1_UP.

3. Do řetězců STANICE1_DOWN respektive STANICE1_UP přidáme výše uvedená pravidla pro směrování paketů do cílových řetězců reprezentujících jednotlivé služby.

4. Ze všech cílových řetězců reprezentujících jednotlivé služby pošleme pakety do řetězce STANICE1_DOWN_ALL respektive STANICE1_UP_ALL

5. Výstup z řetězců STANICE1_DOWN_ALL a STANICE1_UP_ALL nastavíme na požadovanou hodnotu podle uvedeného statusu stanice. V případě statusu „A“ a „WG“ použijeme hodnotu „ACCEPT“. To znamená, že stanice má přístup k vnější síti a budeme pro ní vytvářet statistiky ve formě grafů, respektive ji jen povolíme přístup bez generování grafů. Je-li status nastaven na „N“ použijeme hodnotu DROP. Firewall bude takové pakety zahazovat a blokujeme tak stanici přístup k vnější síti. Poslední možností je nastavení statusu na „JC“ čímž předáme paket dále a o jeho osud se postará jiný řetězec definovaný například při pozdějším přidání pravidel do filtrovací tabulky firewallu.

6. Ze standardně vestavěného řetězce PREROUTING provedeme pro pakety přicházející z vnější sítě na portu uvedeném v tabulce o stanicích (ve sloupci přesměrované porty) překlad adresy na cílovou adresu stanice (DNAT). V opačném směru přeložíme zdrojovou adresu stanice na adresu našeho směrovače (SNAT) v řetězci POSTROUTING.

Tímto způsobem nastavíme strukturu Iptables pro všechny stanice uvedené v tabulce a v případě, že nedojde k žádné chybě, zapíšeme všechny změny filtrovací tabulky najednou zpět do paměti jádra systému. V kapitole 6 Porovnání výkonnosti je pak pozorovatelný nárůst výkonu mezi nastavováním Iptables klasickým a výše uvedeným způsobem. Úspora času je dána tím, že nedochází při každé změně tabulky Iptables k ukládáním změn do paměti jádra systému.

4 Přístup k datům

Jak patrno z modelové představy, aplikace Wsdemon vytváří vrstvu pro přístup k datům o jednotlivých statistikách. Aby byla data přístupná nejen pro lokální počítač, Wsdemon podporuje přenos dat pomocí architektury klient server. To zaručuje klientským aplikacím, umístěným buď na lokální síti nebo kdekoliv v Internetu, přístup k datům poskytovaným serverem. Zároveň je však možné monitorovat jednou klientskou aplikací provoz na několika směrovačích zároveň.

4.1 Klient - Server architektura

Klient server je síťová architektura, která odděluje klienta od serveru. Klientská aplikace je většinou vytvářena tak, aby byla uživatelsky přívětivá a tudíž často využívá grafické rozhraní. Naopak serverová aplikace je v zásadě naprogramována tak, aby nehýřila systémovými prostředky a poskytovala co největší výkon pro klienty. Jedná se tedy pouze o konzolovou aplikaci. Instance klientské aplikace zasílá požadavky na server, který zašle zpět klientovi odpověď.

Z pohledu koncových částí je stejné jestli se klientská a serverová aplikace nachází na stejném počítači nebo v jiné síti. Jak již bylo uvedeno dříve, komunikují spolu aplikační vrstvy z pohledu ISO/OSI modelu a skutečné spojení a vlastnosti spojení zabezpečují vrstvy nižší. Aplikace však musí respektovat typ spojení. V případě spojované služby (typicky TCP protokol) je vytvořena mezi klientskou a serverovou aplikací „roura“, do které jednotlivé konce pouze „sypou“ data. Aplikace se musí postarat o rozlišení jednotlivých požadavků od sebe prostřednictvím určité syntaxe komunikace. V případě nespojované služby (typicky UDP protokol) musíme naopak v aplikaci zabezpečit potvrzování jednotlivých požadavků pro zajištění spolehlivosti.

Podle způsobu realizace servery rozdělujeme na:

- Sekvenční

  Dokáží obsloužit pouze jednoho klienta najednou a ostatní musí počkat ve frontě. Používají se většinou tam, kde výměna dat mezi klientem a serverem je velice rychlá a další klienti nemusí dlouho čekat na akceptování spojení. Jedinou větší výhodou zůstává menší spotřeba paměti a systémových zdrojů než v případě souběžných serverů.

- Souběžné

  Obslouží více požadavků v jednu chvíli, klienti tedy nečekají na vyřízení předchozí žádosti. K obsloužení každého klienta je vytvořen nový proces.

Za výhody této architektury považujeme:

- Veškerá data jsou uložena na serveru, což nám poskytuje větší bezpečnost a snadnější manipulaci. Server může povolit přístup k datům pouze autorizovaným uživatelům.
- Je více flexibilní nežli výměna dat mezi jednotlivými klienty navzájem.
- Klientská i serverová aplikace jsou odděleny a můžeme je vylepšovat nezávisle.
- Rozdělení zátěže. Serverová aplikace není zatížena grafickým zpracováním dat.

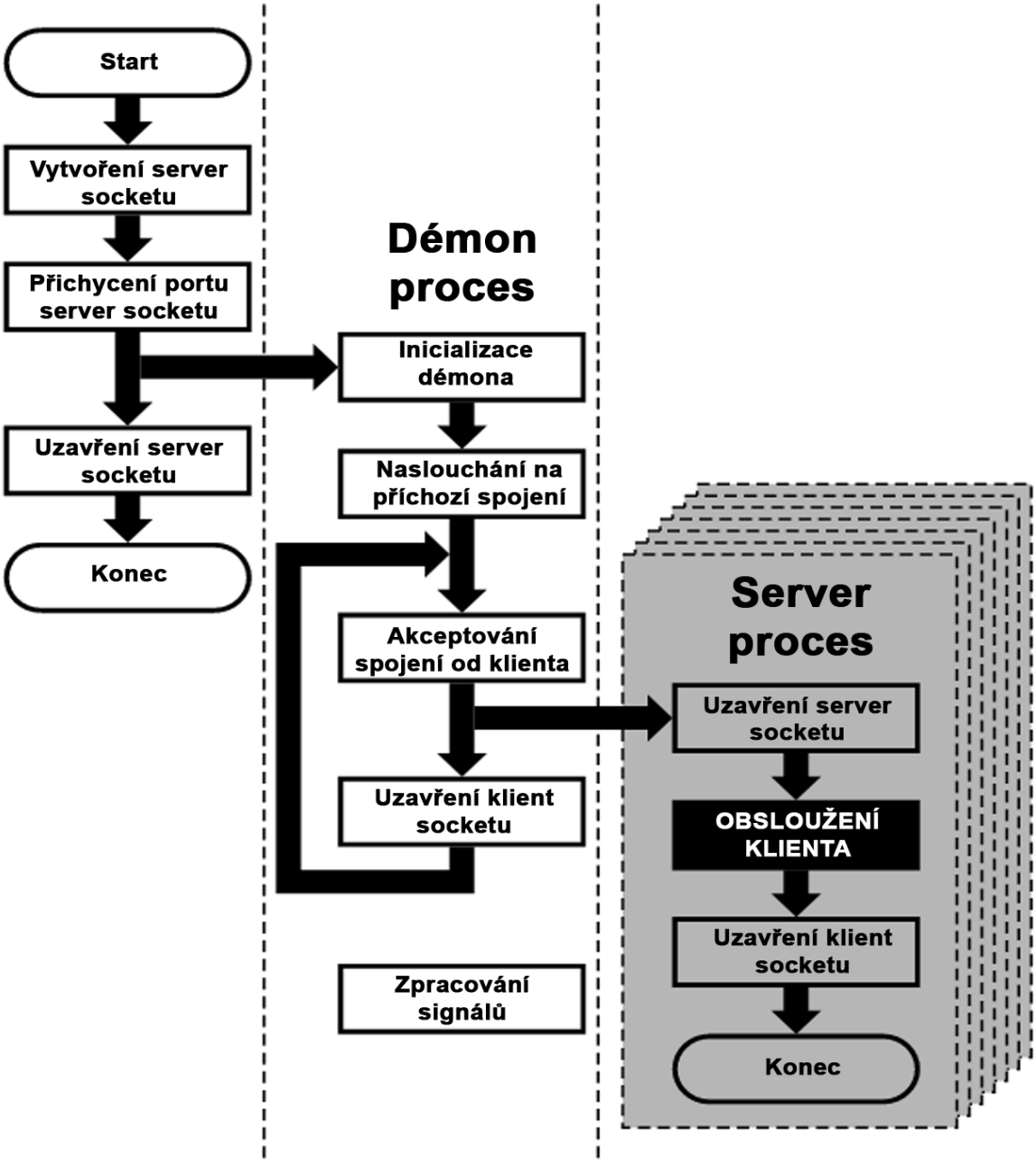
Nevýhody:

- Pokud není spuštěna nebo se zhroutí serverová aplikace, klienti nemají přístup k datům. U výměny dat typu peer to peer toto nehrozí.
- Velké množství požadavků od klientů v jeden moment může zcela zahltit server.

4.2 Realizace Klient - Server

K vlastní realizaci přenosu dat pomocí této architektury jsem využil ukázkové řešení z knihy „TCP/IP Sockets in C: Practical Guide for Programmers“ [7]. Aby server aplikace umožnila současné připojení několika klientů zároveň je použito řešení se souběžným zpracováním požadavků od klientů - tedy souběžný server.

Vývojový diagram zpracování požadavků souběžného serveru je vidět z Obr. 5 - Souběžný server - diagram obsloužení klientů.



Obr. 5 - Souběžný server - diagram obsloužení klientů

Po spuštění serveru se vytvoří socket (soket), který je přichycen k zadanému portu. Hlavní proces předá pouze nově vytvořený soket démon aplikaci a sám se ukončí. Démon proces naslouchá na zadaném portu pro příchozí spojení od klientů. Při připojení klientské aplikace dojde k akceptování spojení a vytvoří se nový proces metodou *„fork“* pro obsluhu samotného klienta, kterému se předá identifikátor soketu. Nově vytvořený proces (potomek) zdědí všechny otevřené sokety od démona (rodiče). Potomek má tedy otevřený nejen soket pro obsluhu klienta, který se vytvořil při akceptování spojení od klienta, ale taktéž hlavní rodičovský soket. Pro obsluhu klienta však rodičovský proces není potřeba a musí být ukončen, aby nedošlo dříve nebo později k vyčerpání popisovačů na soubory (se sokety se zachází stejně jako se soubory). Po dokončení obsluhy klienta je uzavřen i klientský soket.

V Unixu při ukončení procesu nedojde k jeho samotnému zániku, ale pouze k navrácení do místa odkud byl zavolán. To znamená, že při dokončení obsluhy klienta nedojde k ukončení procesu obsluhy, ale pouze k navrácení jeho rodiči, tedy démonovi. Typicky rodičovský proces čeká na návratovou hodnotu jeho potomka. V našem případě se však démon nemůže po vytvoření potomka zastavit a pouze čekat na návratovou hodnotu. To by naprosto popíralo smysl vytvoření dalšího procesu pro obsluhu. Nebude-li však démon čekat na návratové hodnoty svých potomků dojde k tomu, že z ukončených potomků se stávají *„zombies“*. Takto označujeme procesy které již neplní žádnou funkci, ale nejsou zcela ukončeny.

Z tohoto důvodu démon proces potřebuje nastavit manipulaci se systémovými signály v rámci inicializace. Aby mohl démon odebrat návratové hodnoty zombie procesů ze systému a uvolnit tak systémové zdroje, které tyto procesy stále využívají, musíme zpracovávat minimálně signál SIGCHLD. Tuto funkci ve výše zobrazeném vývojovém diagramu reprezentuje blok „zpracování signálů“, který není s ničím propojen.

Samotný server v aplikaci Wsdemon je postaven pro komunikaci na protokolu TCP, abychom se nemuseli starat o pořadí a potvrzování zpráv. K tomu abychom mohli rozlišit jednotlivé dotazy, které postupně vysíláme na druhý konec, musíme definovat určitou syntaxi komunikace, která bude pevně dána. K rozlišení začátků a konců příkazů nám postačí definovat ukončovací znak, podle kterého klient či server pozná, že se jedná o poslední znak daného dotazu. Jelikož různé programovací jazyky použité pro klientské aplikace používají ke čtení z proudu dat různé ukončovací značky, definoval jsem syntaxi pro komunikaci tak, aby byla jednotná pro všechny klienty.

Dotazy od klienta na server jsou rozpoznávány serverem napsaném v jazyku C a stačí tedy dotazy ukončovat symbolem NULL *„\0“*, tedy prvním znakem z ASCI tabulky. V opačném směru, od serveru ke klientům, je použit ukončovací řetězec *"\n\0"*.

Server podporuje tyto dotazy:

- „authorization,heslo“ - žádost o autorizaci pomocí hesla zakódovaného šifrovacím algoritmem MD5
- „get_clients“ - žádost o zaslání jmen klientských stanic
- „get_clientinfo,jmeno“ - žádost na informace o klientské stanici, kde jméno reprezentuje jméno klientské stanice
- „get_clientcounters,jmeno“ - požadavek na zaslání hodnot počítadel pro všechny přednastavené služby v obou směrech
- „get_counter,jmeno“ - požadavek o zaslání hodnoty počítadla jedné statistiky
- „get_clientspeeds,jmeno“ - požadavek na zaslání rychlostí dat klientské stanice procházející jednotlivými počítadly pro všechny přednastavené služby v obou směrech
- „get_services“ - dotaz na jména všech přenastavených služeb
- „bye“ - oznámení ukončení komunikace ze strany klienta

Mezi výhody, které k takovémuto přístupu k datům patří:

- možnost získávat informace ze směrovače odkudkoliv z vnitřní sítě či Internetu
- směrovač nemusí být nijak výkonná stanice, protože veškeré grafické zpracování dat může probíhat na jiném počítači (generováním grafů nezatěžujeme směrovač)
- jedna klientská aplikace sloužící pro vizualizaci může zpracovávat statistiky z více směrovačů najednou
- rychlejší přístup k datům o jednotlivých statistikách z Iptables než v případě parsování výstupu

Při zavolání aplikace Wsdemon aplikace s parametrem „*-R port*“, kde port reprezentuje číslo portu dojde ke spuštění souběžného serveru na daném portu TCP protokolu. Jak vypadá samotná komunikace se serverem je zobrazeno na Obr. 6 - Výpis komunikace na serveru.

```
============ Handling client [98] with child process: 0 ==============


Received authorization,9ebccc5f5c80b6ff1a4660baf9c5ed35
COMMAND:"authorization"

Send [23]: Authorization success!


Received get_clients
COMMAND:"get_clients"

Send [106]: WITR2,WITRNOTAS,WITRNOTAS2,WITR,ALES,ALSNOTEBOOK


Received get_clientinfo,WITR
COMMAND:"get_clientinfo"

Send [71]: CLIENTINFO;WITR;10.1.1.2;00:60:08:6A:5C:5E;3968;976;3912;912;1;none,;A



COMMAND:"get_services"

Send [56]: SERVICES;9;HTTP,FTP,P2P,MAIL,SSL,REMOTE,AUDIO,VIDEO,OT,


Received get_clientspeeds,WITR
COMMAND:"get_clientspeeds"

Send [44]: SPEEDS;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;


Received get_clientspeeds,WITR
COMMAND:"get_clientspeeds"

Send [59]: SPEEDS;135995;15704;0;0;0;0;0;0;0;0;0;0;0;60379;821;0;0;


Received get_clientspeeds,WITR
COMMAND:"get_clientspeeds"

Send [60]: SPEEDS;349462;27949;0;0;0;0;0;0;0;0;0;0;0;88200;1136;0;0;


Received get_clientspeeds,WITR
COMMAND:"get_clientspeeds"

Send [61]: SPEEDS;450114;8481;0;0;0;0;0;0;0;0;0;0;0;54444;800;301;40;
```

Obr. 6 - Výpis komunikace na serveru

4 Vizualizace statistik

Pro grafické zpracování dat je zapotřebí použít různých technologií v závislosti na časovém rozmezí, které chceme graficky zobrazovat. Dlouhodobé statistiky v řádu hodin až několika let se zpracovávají ve skriptu vytvořeném v jazyce PERL s využitím opensource softwaru „RRDTool“. RRDtool využívá vlastní databázový systém pro ukládání dat v cyklech. Podrobné informace o RRDTOOL nalezneme na stránkách autora [8].

Pro vizualizaci aktuálně protékajících dat přes náš směrovač byl vytvořen skript v jazyce PHP s využitím rozšiřujících balíčků „PEAR“.

Oba tyto klienti generují v pravidelných intervalech data ve formě grafů na webový server.

4.1 Klientská aplikace pro dlouhodobé statistiky

Pro vizualizaci statistik na webovém serveru je nutno realizovat tyto kroky:

1. Přenos dat mezi klientskou a serverovou částí

2. Uložení proměnných reprezentujících jednotlivá počítadla do databáze

3. Vygenerování grafů z hodnot uložených v databázi

4. Automatické vygenerování HTML stránek obsahujících jednotlivé grafy

Pro tento účel byl naprogramován skript v jazyce PERL, který je automaticky spouštěn pomocí „*cron*“ démona každé 2min. Při každém spuštění skriptu *„Wspclient.pl csg“* dojde k přenosu dat ze serveru zadaného IP adresou a portem, uložení hodnot do RRDTool databáze, překreslení grafů, které je nutno aktualizovat, do souboru ve formátu „*png*“ a vygenerování HTML stránky zobrazující jednotlivé grafy.

ad 1) Přenos dat

Pro přenos dat se musí nejprve klientská aplikace autorizovat zasláním požadavku o autorizaci s heslem zakódovaným pomocí šifrovacího algoritmu MD5. Heslo je šifrováno proto, aby neputovalo v síti v čitelné podobě. Zašifrovaný řetězec se na straně serveru porovná se vzorem uloženým v konfiguračním souboru a v případě shody klient může pokračovat v komunikaci. Nezašle-li klient jako první dotaz na server žádost o autorizaci nebo pošle žádost se špatným heslem je automaticky odpojen, se sdělením o nevydařené autorizaci. Samotné data jsou pak zasílána ve tvaru řetězce rozděleného středníky pro snadné rozparsování v klientských skriptech.

ad 2) RRDTOOL Databáze

Pro jednotlivé stanice v síti je vytvořen soubor s pevnou délkou ve formě cyklické databáze, která obsahuje datové položky reprezentující jednotlivé statistiky. Ze všech hodnot vypočítáme pomocí konsolidační funkce průměry vstupujících dat odpovídající intervalům 2 minuty (1 data point pro průměr), 10 minut (5 data pointů) a 30 minut (15 data pointů), které jsou pak použity k tvorbě.

Aktualizace hodnot do databáze probíhá každé 2 min. V případě, že data nepřijdou do 4 minut od poslední aktualizace a nebude k dispozici více jak 50% hodnot pro daný průměr, budeme považovat daný průměr za neznámý. Taktéž v případě, že vstupující data budou mimo interval <0;100Mbit>, což odpovídá množině možných hodnot rychlostí na síťovém rozhraní daného směrovače.

ad 3) RRDTOOL grafy

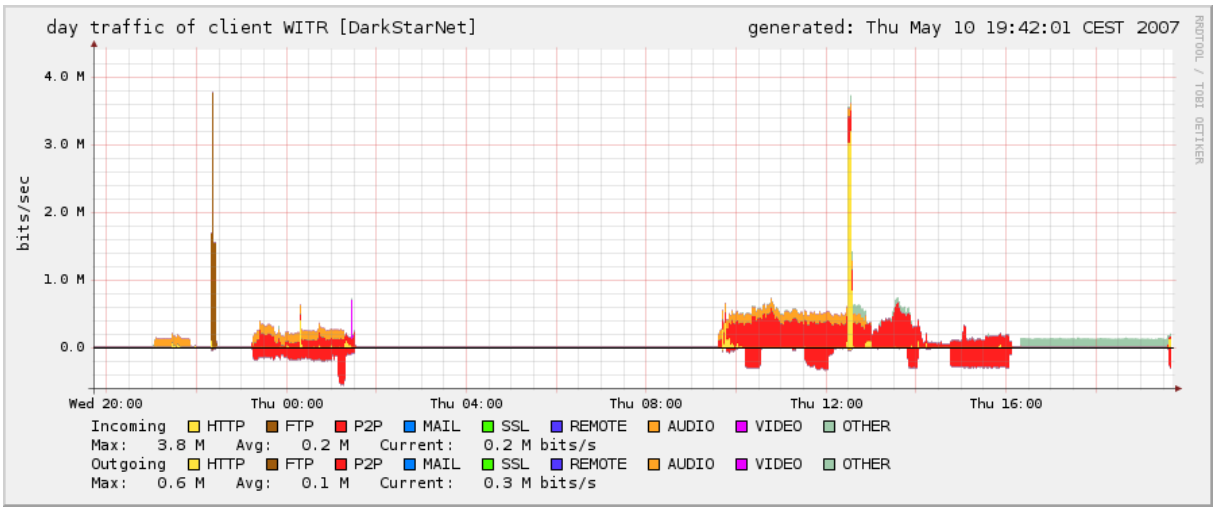
Zároveň při každém spuštění skriptu provádíme aktualizaci grafů (tedy pouze těch, které je nutno aktualizovat, což odpovídá parametru „*--lazy*“). Pro generování grafů přepočítáváme hodnoty z RRA:AVVERAGE, které obsahují hodnoty v bytech, pomocí reverzní polské notace na hodnoty v bitech. Směr provozu od stanice do Internetu přepočítáme do záporných hodnot, tak aby se v grafu zobrazoval pod horizontální osou.

ad 4) RRDTOOL grafy

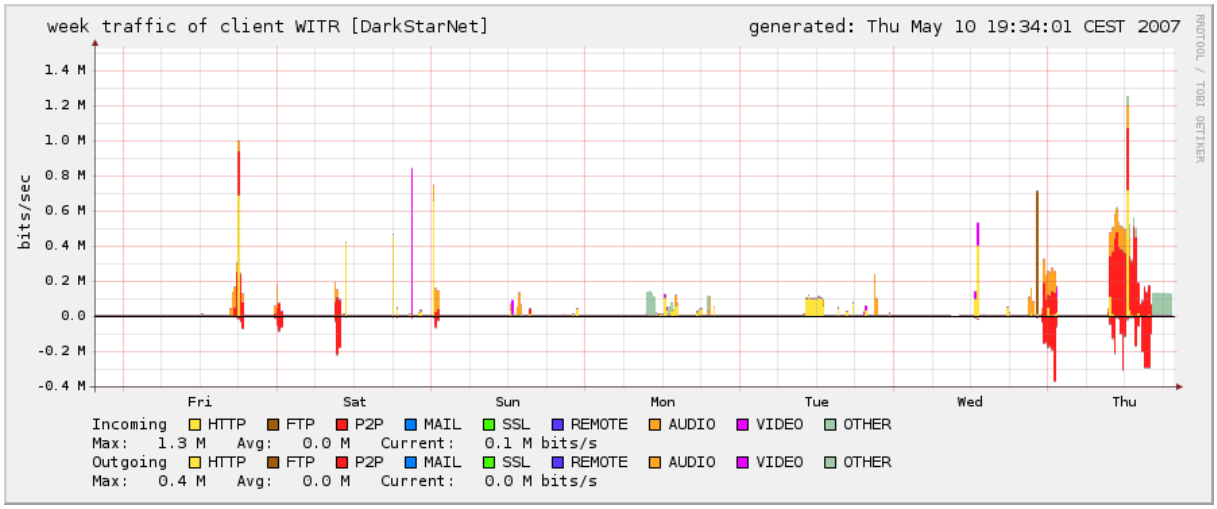
Aby nebylo při přidání/odebrání stanice z tabulky sloužící jako vstup aplikace Wsdemon nutné měnit kód HTML stránky zobrazující grafy, je ve skriptu vytvořen také generátor HTML stránek. Na webovém serveru je zobrazen provoz sítě v rámci jednotlivých služeb v různých časových intervalech: denní, týdenní, měsíční a roční.

Ukázky dlouhodobých grafů

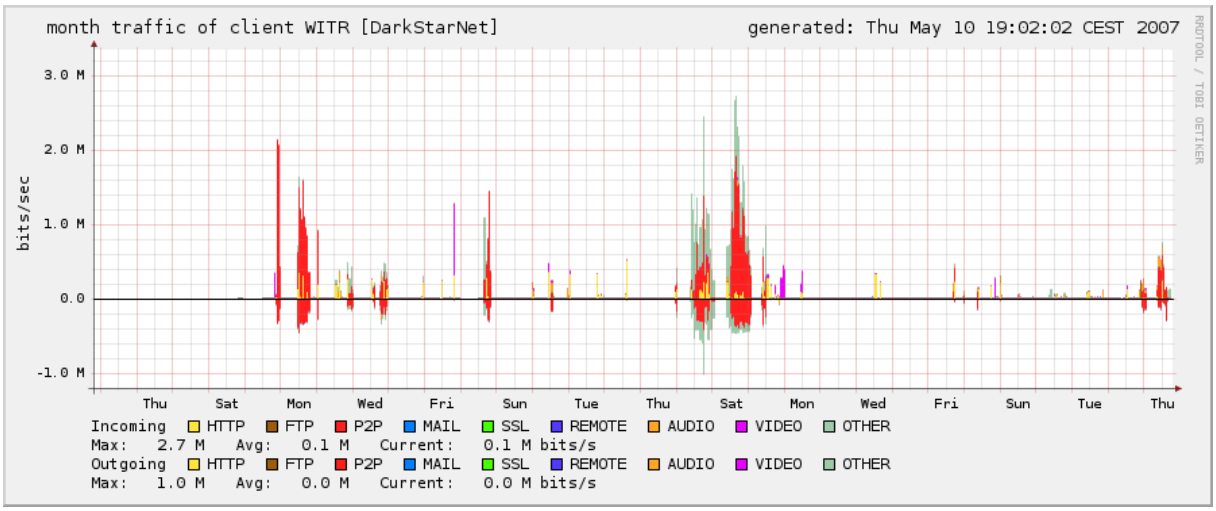
V grafu je uvedeno vždy jméno stanice, do jaké sítě tato stanice patří, datum kdy byl graf vygenerován, maximální, průměrné a aktuální hodnoty rychlostí přenosů v obou směrech. Směr přenosu dat z Internetu k dané stanici je zobrazen nad osou x, směr od stanice do Internetu pod osou x v záporných hodnotách. Barevně jsou rozlišeny přenosy na jednotlivých službách.



Obr. 7 - Graf denní statistiky

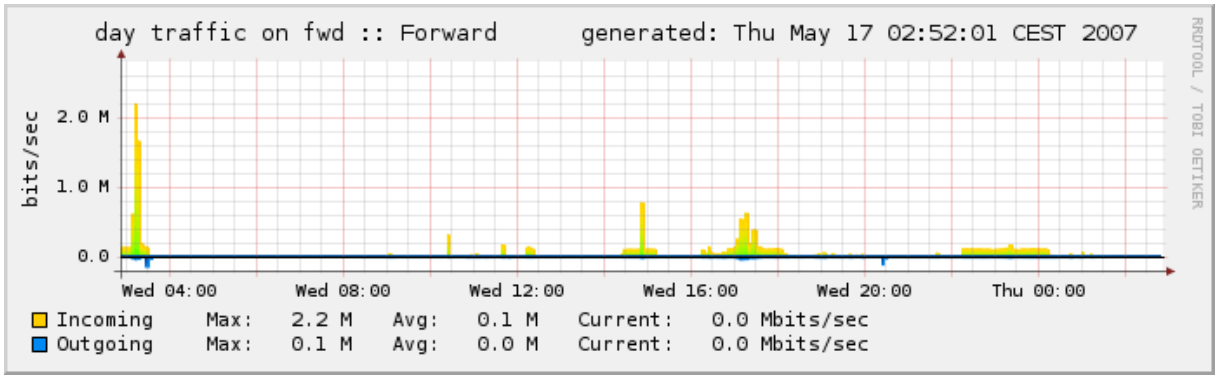


Obr. 8 - Graf týdenní statistiky



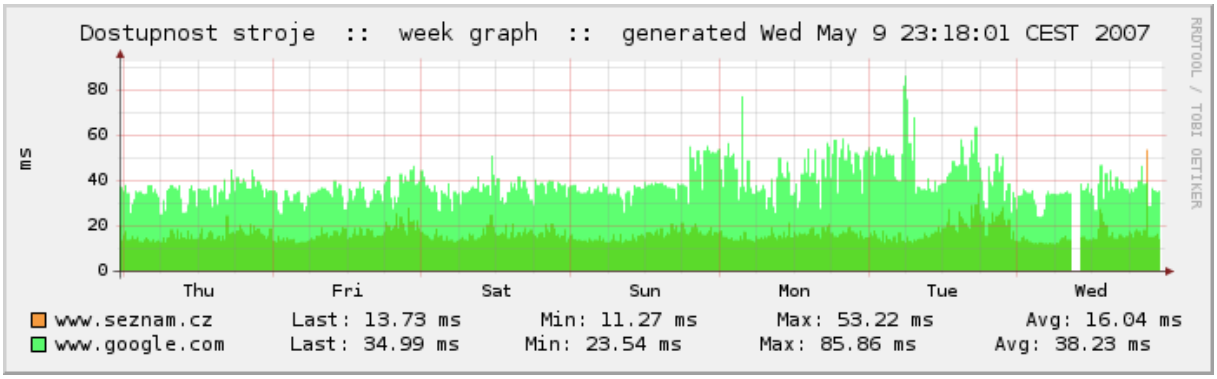
Obr. 9 - Graf měsíční statistiky

Dále je vytvářena statistika celkového provozu mezi vnitřní a vnější sítí, viz. Obr. 10 - Celkový provoz mezi vnitřní a vnější sítí. Datový provoz směrem z vnější do vnitřní sítě (download) je nad x-ovou osou, v opačném směru (upload) je zobrazen pod osou.



Obr. 10 - Celkový provoz mezi vnitřní a vnější sítí

Zároveň s ostatními je generován graf dostupnosti sítě na Obr. 10 - Dostupnost sítě, který zobrazuje dostupnost sítě a odezvu na zvolené servery v Internetu.



Obr. 11 - Dostupnost sítě

4.2 Klientská aplikace pro krátkodobé „realtime“ statistiky

Informace jsou vizualizovány pomocí skriptu v jazyku PHP s využitím rozšiřujícího modulu „PEAR-Net_Sockets“ pro přenos dat a „PEAR-Image_Graph“ pro generování grafů.

Při vizualizaci pomocí skriptu jakožto webové aplikace si musíme uvědomit rozdíl mezi zobrazováním grafů v desktopové aplikaci nebo jako součásti webové stránky. V desktopové části při zobrazování grafu jsou použity komponenty umožňující překreslování jednotlivých částí grafu s podporou rychlých knihoven a uživatel na průměrném počítači nepostřehne okamžik, kdy došlo k překreslení. V případě webu nemáme příliš na výběr a grafy jsou zobrazovány na pravidelně obnovované webové stránce jako vygenerované obrázky. Vygenerování takového obrázku je výpočetně poměrně náročná operace a trvá řádově stovky ms až sekundy v závislosti na výkonnosti počítače.

Klasické webové technologie pracují na synchronním principu, tedy nejprve klient zašle žádost na webový server o určitou stránku, kterou chce zobrazit a server poté pošle požadované data zpět klientovi. Tato data se pak zobrazí v klientském prohlížeči. V čase mezi zasláním žádosti o webovou stránku na server a přijetím nových dat nemá klientský prohlížeč k dispozici žádná aktualizovaná data. Každý prohlížeč si s touto situací poradí jinak. Některé z nich nechají uživatele čekat s prázdnou stránkou, jiné zachovají obsah stránky až do doby, kdy dojde k přijetí nových dat.

Zvolíme-li obnovovací interval stránky například sekundu a generování grafu ve formě obrázku trvá taktéž sekundu zjistíme, že polovinu času uživatel vidí prázdnou stránku s přesýpacími hodinami, což je přinejmenším mrzuté.

Na webové stránce s grafem můžeme mít další prvky, které není potřeba opakovaně obnovovat. Při obnovení celé stránky načítáme zbytečná data navíc a celý proces se tím zpomaluje.

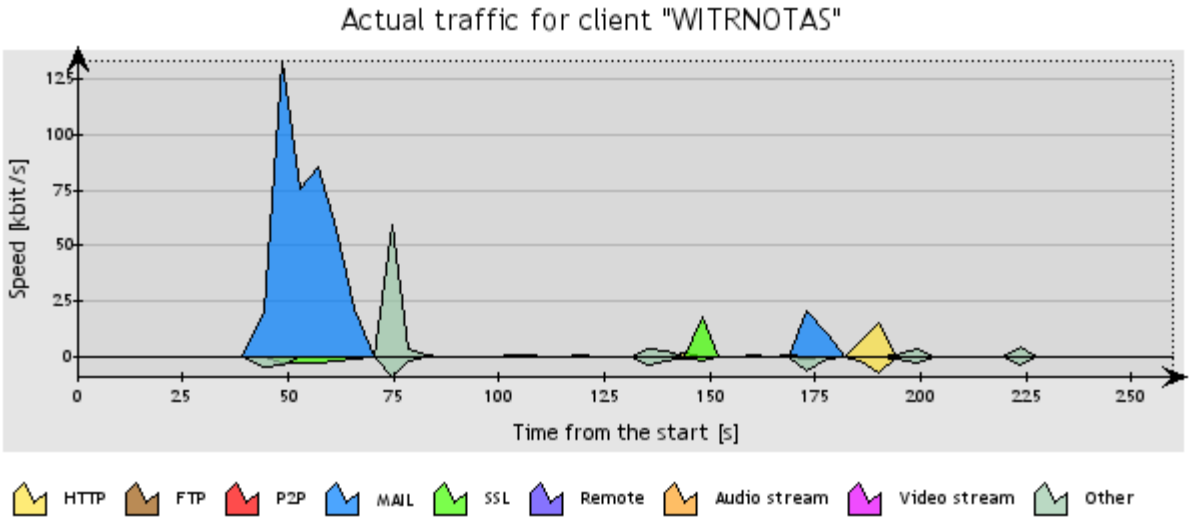
Jako současné řešení pro odstranění těchto problémů se jeví použití technologie *„AJAX“*, kterou jsem i já využil k vizualizaci realtime statistik. AJAX je obecné označení pro technologie vývoje interaktivních webových aplikací. Mezi výhody patří odstranění nutnosti opakovaného načítání a překreslení celé stránky při každé operaci. Další výhoda AJAXU je schopnost komunikovat a zobrazovat stránku v klientském prohlížeči asynchronně. To znamená, že jsou použity signály v javascriptu, které informují o tom v jakém stavu je aktuální požadavek na server a k překreslení stránky dojde až v okamžiku když je již celá stránka načtena v paměti.

Další problém, který jsem musel řešit, bylo zachování otevřeného spojení na server mezi jednotlivými obnovami stránky. Jelikož webové aplikace fungují na principu skriptů, které mají daný časový limit, do kterého musí být kompletně zpracovány, není z principu možné toto spojení zachovávat. Pokusy ukládat veškeré informace o soketu do „session" proměnné a znovu je načítat skončily neúspěchem, jelikož v momentě kdy načteme informace o socketu zpátky ze session, bylo spojení již uzavřené. Při každé obnově stránky tedy dochází k synchronizaci spojení mezi serverem a klientem, opakované autorizaci a až poté k samotnému přenosu informací.

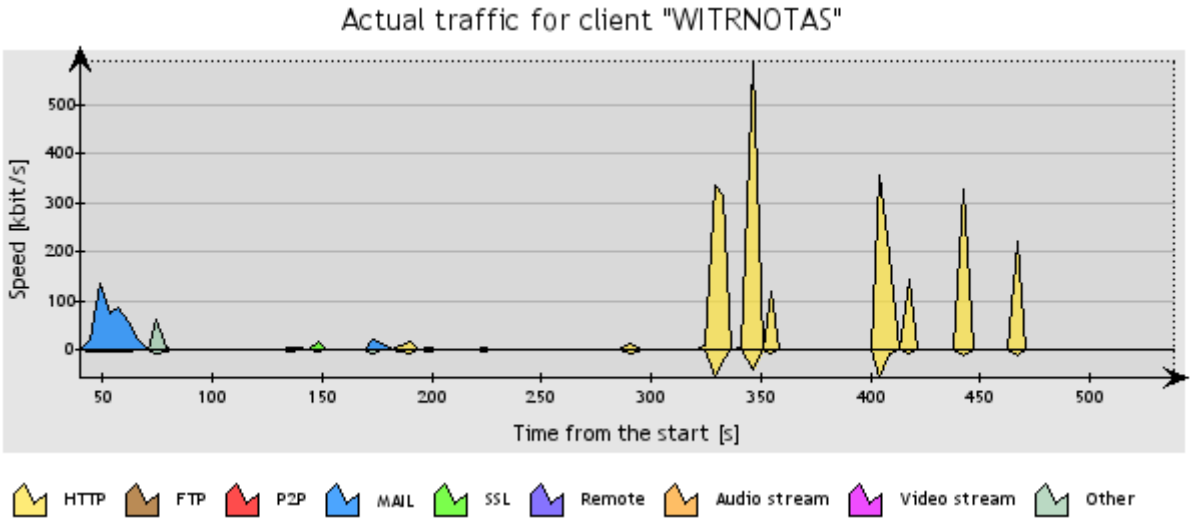
Server Wsdemon sice podporuje požadavky přímo na hodnoty aktuálních rychlostí jednotlivých statistik, ale bohužel v rámci jednoho otevřeného spojení. To vyplývá z architektury souběžného serveru, kdy k obsloužení jednoho klienta je vždy vytvořen nový proces. Daný proces pak zachovává v paměti informace o posledních hodnotách počítadel a podle uplynulého času a rozdílu předchozích a nových hodnot spočte aktuální rychlosti. První hodnota rychlosti je tedy vždy nulová. V případě, že se klientská aplikace při každém dotazu připojí, získá data a hned odpojí, získá vždy pouze nulové hodnoty rychlostí.

Aktuální rychlosti jednotlivých statistik jsou proto vypočítávány z absolutních hodnot počítadel až v samotném skriptu a ukládány do session proměnné, odkud jsou při každém obnovení stránky zpět načítány. Hodnoty časů na x-ové ose pak reprezentují skutečné časy kdy byla data získána.

Ukázka grafů realtime statistik pro jednu stanici je zobrazena na obrázcích dále. Délka časového okénka odpovídá počtu datových bodů na x-ové ose vynásobených obnovovací frekvencí stránky. Nejprve dochází k plnění grafu novými hodnotami směrem zleva, což odpovídá Obr. 12 - Realtime statistika stanice a při naplnění celého okénka hodnot dojde k posouvání grafu do leva – viz. Obr. 13. Barvami jsou odlišeny datové toky jednotlivých aplikačních protokolů. V kladných hodnotách jsou pak vyneseny přenosy dat „downloadu" v kilobitech za sekundu (kbit/s) směrem z vnější sítě (Internetu) do vnitřní sítě (LAN), v opačném směru jsou vyneseny hodnoty pod osou x a reprezentují „upload".

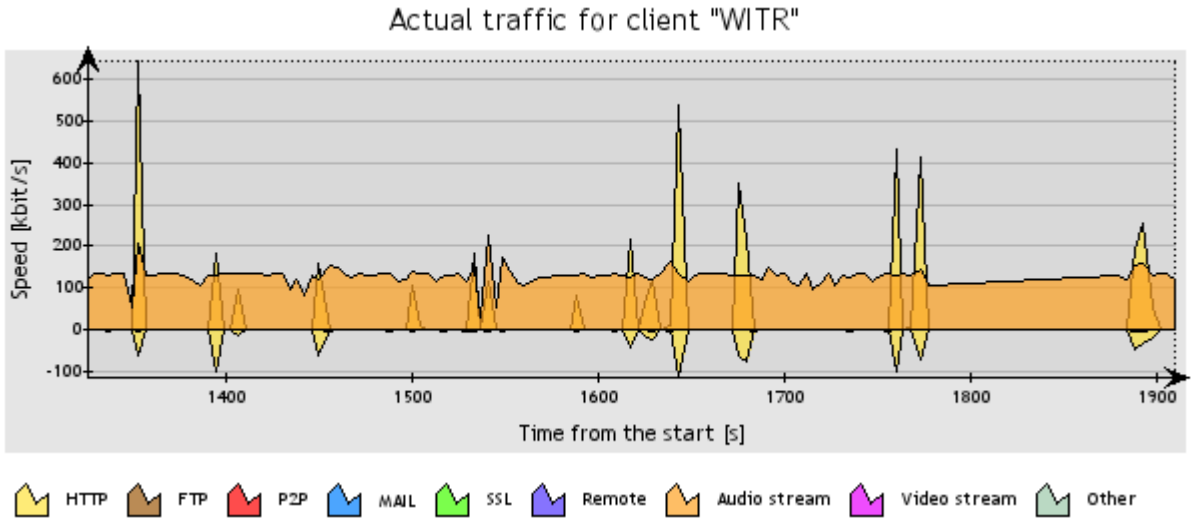


Obr. 12 - Realtime statistika stanice



Obr. 13 - Realtime statistika stanice

Na Obr. 14 - Realtime statistika - audio stream a web je zobrazen průběh datových přenosů klientské stanice, na které měl uživatel spuštěné internetové rádio v kvalitě 128kbit/s a používal www službu.



Obr. 14 - Realtime statistika - audio stream a web

5 Zajištění kvality služby

5.1 Úvod do problematiky Quality of Service (QoS)

V poslední době došlo k velkému zvýšení počtu účastníku internetu, což s sebou přináší problémy se zajištěním kvality jednotlivých služeb. Zajištění kvality musíme chápat ve vztahu ke způsobu služby, některé přenosy vyžadují maximální propustnost, jiné zase minimální dobu odezvy. Organizace ITU-T definuje ve svém doporučení E.800 kvalitu služby (QoS) jako „*souhrnný efekt výkonnosti služby, který určuje stupeň uspokojení uživatele této služby"*. Pro praktickou implementaci v aktivních prvcích sítě je zapotřebí definovat přímo kvantitativní parametry:

- propustnost sítě pro datový tok
- zpoždění dat na trase mezi zdrojovou a cílovou stanicí

Propustnost sítě je dána jak kvalitou linek (kapacitou a chybovostí) tak i parametry aktivních prvků podél celé trasy, zejména jejich výpočetním výkonem a velikostí vyrovnávacích pamětí.

U zpoždění dat jsou důležité statistické parametry, střední hodnota a rozptyl. Pro jednosměrné proudy dat (např. video stream televizní stanice) není příliš důležitá samotná hodnota zpoždění, avšak velký rozptyl může způsobit pozorovatelné zhoršení kvality příjmu. Pro interaktivní aplikace jako je videokonference, VOIP, ale též některé grafické aplikace, jsou velmi podstatné oba parametry. Na kvalitu služby mají vliv všechny uzly sítě mezi komunikujícími stanicemi.

Existuje více modelů implementace QoS. Model nazývaný „IntServ" předpokládá záruky pro QoS po celé trase mezi zdrojovou a cílovou stanicí. Všechny uzly na trase musí získat informace o parametrech požadovaných pro daný datový tok a vyhradit pro něj odpovídající zdroje, které stojí v případě sítě velikosti Internetu nemalé režie na každém aktivním prvku.

Podstatně jednodušší model nazvaný „DiffServ" je založen na agregaci datových toků do malého počtu tříd (Class of Service - CoS) a jim odpovídajících kvalitativních typů služeb „ToS". V každé hlavičce paketu je pak na hraničním směrovači mezi klienty a ISP možné parametr ToS nastavit. Směrovače v jádru sítě (nehraniční) již komunikují režimem best effort.

5.1.1 Součásti QoS

Je zřejmé, že efektivní implementace QoS musí zahrnovat následující funkce:

1. Realizace administrativní strategie
2. Klasifikace paketů
3. Řízené odesílání paketů

Ad 1) Je třeba kontrolovat, zda případné rezervace zdrojů nepřekračují dostupnou kapacitu. Součástí strategie je i způsob jakým se nakládá s požadavky či datovými toky, které neodpovídají uvedeným předpokladům. Tedy například, zda nezpůsobilé pakety zahazovat nebo je pouze převést na „best effort“[1], zda lze dodatečně omezit existující datový tok v případě, že se objeví požadavek s vyšší prioritou a podobně.

Ad 2) Díky klasifikaci paketů procházejících přes směrovač můžeme nastavit různým službám různé parametry. Například pro interaktivní komunikaci při vzdálené chirurgii nastavíme vyšší prioritu v síti než přenosu dat pomocí FTP. Z výstupu klasifikátorů vystupuje datový tok paketů označený značkami.

Ad 3) Pro určení jakým způsobem bude s daným paketem datagramu naloženo slouží fronty paketů. Fronta paketů je pořadí, v jakém budou pakety odeslány ze síťového rozhraní. Způsob řazení do této fronty je právě algoritmus pro zajištění kvality služeb.

---

1 Datové komunikace v Internetu se stále a téměř bez výjimky realizují v režimu best effort (BE)-“Maximální snaha”. Tím se myslí, že přenosové části sítě by měly především přenášet data, a to co možná nejlépe, a neměly by se naopak zdržovat jinými věcmi, mezi které patří zejména zajištění spolehlivosti, neboli napravování případných chyb, pokud k nim při přenosech došlo. Přenosové mechanismy se samozřejmě snaží přenášet data bez chyb a vyvarovat se aktivit, které by mohly k chybám vést. Jakmile ale již jednou k nějaké chybě dojde, nepovažují za svou povinnost postarat se o nápravu. Díky tomu pak mohou fungovat efektivněji, protože nemusí nést režii nezbytně spojenou s případným zajišťováním spolehlivosti. Protokol IP (Internet Protocol), který je hlavním přenosovým protokolem TCP/IP (a funguje na úrovni síťové vrstvy) je protokolem, který funguje na právě popsaném nespolehlivém principu.

Je důležité si uvědomit, že můžeme ovlivnit pouze data, která vysíláme. Díky způsobu, jakým funguje Internet, nemáme žádnou přímou kontrolu nad tím jaká data k nám proudí.

Příchozí provoz na síťovém rozhraní se označuje „ingress“ a jeho tvarování (přeuspořádání, zpomalení, či zahození paketu) se nazývá „policing“. V Linuxu lze na ingress uplatnit pouze omezení šířky pásma pomocí zahazování paketu, například pomocí regulátoru IMQ. Díky vlastnostem protokolu TCP můžeme zahazovat pakety, pokud chceme docílit snížení rychlosti přijímaných dat. Toto se ale netýká jiných protokolů, kde zahazování paketů nemá na rychlost žádný vliv. Z principu nelze použít žádné pokročilejší způsoby klasifikace.

Odchozí provoz se označuje jako „egress“ a jeho tvarování se nazývá „shaping“. Pomocí shaperu lze pakety zpomalovat, přeuspořádávat i zahazovat za použití různě složitých algoritmů.

5.2 Koncepce QoS v Linuxu

Pro realizaci QoS v Linuxu slouží balíček „tc“ (traffic controller). Základním principem implementace jsou 3 kategorie objektů:

1.Typ fronty (queueing discipline)

Queueing discipline (dále jen disciplína) je přiřazena výstupnímu rozhraní. Disciplína definuje způsob obsluhy a parametry příslušné fronty paketů určených k odeslání.

2.Třída (class)

Třídy jsou sjednocení datových toků, které mají určité společné charakteristické rysy. Umožňují definovat společný způsob zpracování pro tyto toky v rámci nadřazené disciplíny. Třídy obsahují vnořené disciplíny, protože samy nedokáží manipulovat s pakety.

3.Filtry pro klasifikaci

Filtry jsou svázány s třídami a disciplínami. Umožňují klasifikování paketů a jejich rozdělení do tříd a disciplín.

Každá třída musí obsahovat vnořenou disciplínu pro skutečnou manipulaci s frontou. Pokud není specifikováno jinak, jde o disciplínu typu FIFO. Tyto fronty jsou až na výstupu zařízení, to znamená, že nemůžeme omezovat rychlost na vstupu, ale pouze na výstupu ze síťového rozhraní.

5.3 Realizace řízení provozu

Pro praktickou implementaci QoS systému na hraničním směrovači tvořící bránu do Internetu je nutno brát zřetel na tyto věci:

- Implementace QoS pomocí modelu DiffServ nezaručuje v podstatě žádné výsledky. Všechny směrovače, kterými paket prochází od zdroje k cíli nemusí respektovat nastavení pole ToS v hlavičce paketu. Pro ostatní datové přenosy z jiných sítí mohou být ToS parametry nastaveny špatně.

- Efektivně tedy můžeme rozdělovat pásmo mezi jednotlivé služby a stanice v rámci vnitřní sítě.

Principem funkčnosti QoS systému realizovaného v diplomové práci je omezení datových toků stanic, které příliš vytěžují konektivitu do vnější sítě (Internetu). Tento systém se označuje „Fair User Policy“ (FUP) a zajišťuje spravedlivé rozdělení konektivity mezi stanicemi vnitřní sítě.

K implementace FUP do jádra systému jsem použil typy front HTB a SFQ.

HTB - Hierarchical Tocken Buckem

Rozděluje dané přenosové pásmo mezi jednotlivé třídy. Je možné nastavit minimální (rate) a maximální rychlost třídy (ceil), priority mezi třídami (prio). Třídy si mezi sebou mohou půjčovat volné pásmo.

SFQ - Stochastic Fair Queuening

SFQ implementuje oddělení datových toků. Pakety jsou zařazeny do jedné z 256 front a ty jsou pak cyklicky obsluhovány tak, že z každé fronty je vybráno stejné množství dat.

Parametry v HTB jsou definovány následovně:

rate - minimální rychlost třídy
ceil - maximální rychlost třídy
prio - priorita třídy
burst - maximální počet odeslaných dat ze síťového rozhraní pro danou třídu než dojde k obsloužení další třídy

Data, která proudí přes směrovač můžeme pomocí HTB rozdělit do tříd. Každé třídě nastavíme maximální rychlost, minimální garantovanou rychlost a prioritu vůči ostatním třídám. Strukturu tříd vytvoříme tak, aby pro každou stanici ve vnitřní síti byla vytvořena samostatná třída, do které nasměrujeme pomocí filtrů datový tok dané stanice. Parametry, které jsou třídě přiděleny jsou získány z aplikace Wsdemon. Celková struktura je poněkud komplexnější a je podrobněji nastíněna v příloze A.

Rate parametr nastavujeme zhruba jako podíl maximální rychlosti připojení v daném směru ku počtu uživatelů v síti. Ceil je pak maximum rychlosti stanice v daném směru. Parametry Ceil, Prio získáme dotazem „get_clientinfo“ na informace o dané stanici na server Wsdemona. Pro zakázané protokoly je vytvořena velmi pomalá třída, do které dané pakety nasměrujeme. Nastavení QoS do jádra systému se provádí spuštěním skriptu Wspclient s parametrem „*-shaper*“.

Výpočet parametrů pro FUP v aplikaci Wsdemon

Wsdemon využívá k ukládání dat o stanicích jednoduchého textového souboru, který má následující strukturu:

stanice1	10.1.1.2	00:A0:C0:76:2A:F2	2048	512	256	256	2	directconnect		10205	A
stanice2	10.1.1.2	00:A0:C0:76:2A:F2	2048	512	256	256	1	none	none	A	
stanice1	10.1.1.2	00:A0:C0:76:2A:F2	1024	256	128	128	1	none	none	A	

Na jednotlivých řádcích máme tabulátorem oddělené údaje o stanicích v síti v pořadí jméno, IP, MAC adresa, maximální rychlost download, upload, garantovaná rychlost download, upload, priorita v síti, zakázané protokoly oddělené čárkou, přesměrovaný port ke stanici a status v síti.

K nastavení HTB máme tedy všechny údaje k dispozici v této tabulce. Minimální garantovaná rychlost, priorita v síti a zakázané protokoly jsou dané, ke správné funkci FUP je však třeba přepočítávat maximální rychlost pro danou stanici v obou směrech. Já jsem pro přepočet maximální rychlosti zvolil následují postup.

Standardně definované služby HTTP, FTP, P2P, Mail, SSL, Remote, Audio, Video, Other jsou rozděleny do 3 skupin L0, L1, L2. Skupiny mají definovaný koeficient KS, který udává jak mnoho chceme danou skupinu služeb limitovat. L0 nelimitujeme téměř vůbec naopak L2 nejvíce. Zvolil jsem rozdělení služeb takto:

L0 - HTTP, Mail, Remote (ssh,telnet)	KS ~1
L1 – FTP, SSL, Audio, Video, Other	KS ~0.7
L2 – P2P	KS ~ 0.2

Jelikož na síti vznikají špičky, kdy bývá síť vytížená nejvíce a útlumy, kdy je přenosová kapacita sítě nevyužitá, budeme brát také v úvahu aktuální čas na přepočítávání hodnot. K tomu slouží 3 časové zóny, které nadefinujeme pomocí intervalů (TIMEZONES). Jednotlivým intervalům pak přiřadíme koeficient času KT:

```
TIMEZONES="8,16,23";
KT="0.8,1,0.2";
```

Určitě budeme chtít brát v úvahu datový limit, který je možno nastavit zvlášť pro příchozí a odchozí data. Tento datový limit si pak administrátor sítě experimentálně upraví podle kladených požadavků.

Celkově tedy k ovlivnění chování FUP nastavujeme:

- Koeficienty skupin služeb KS = <0;1>
- Časové koeficienty KT= <0;1>
- Koeficient datového limitu DL= <0; nekonečno>
- Časovou prodlevu mezi aktualizacemi

Tyto parametry jsou nastavitelné v souboru wstat.config, který je standardně umístěný s tabulkou uživatelů v adresáři „/etc/wstat“. Konfigurace tohoto souboru je popsána v sekci 7.2 Konfigurace aplikace Wsdemon. Časovou prodlevou mezi koeficienty myslíme jak často budeme vypočítávat nové hodnoty. Ta je dána tím, jak často naplánujeme v „cronu“ démonu jednotlivé aktualizace hodnot. Spuštěním aplikace s parametrem „*Wsdemon -NS*“ vypočítáme nové parametry, které se uloží do souboru.

Samotný výpočet je prováděn následovně:

NSX=(Actual_Max_Speed – Actual_Max_Speed * KT * DD * (1 – KS))
NSX je nová maximální rychlost služby X (HTTP, FTP, P2P, …)

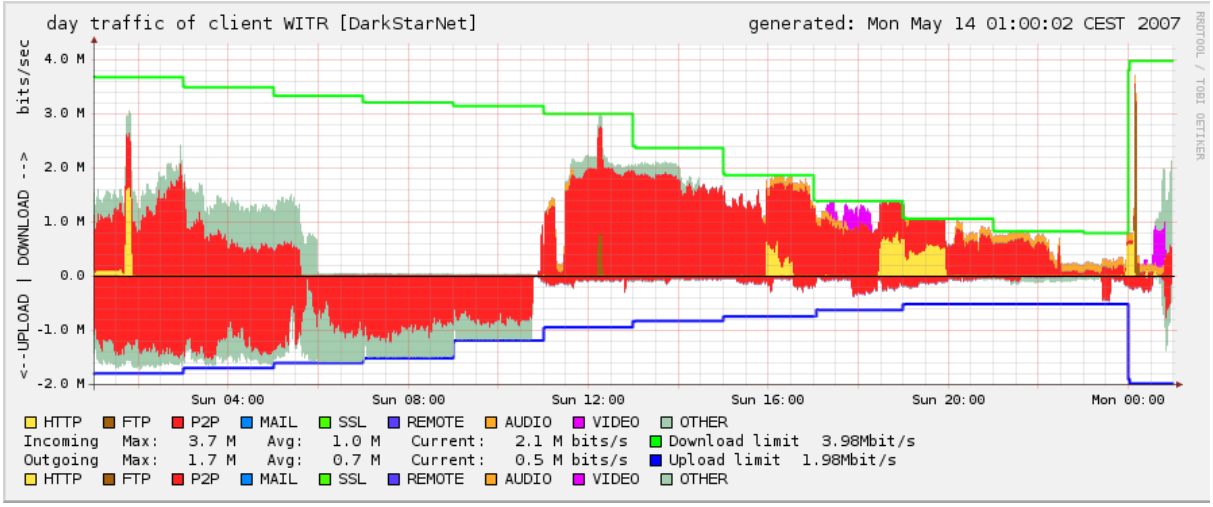
DD= Service_Data_Difference / (Service_Data_Difference + DL)
DD je parametr vznikající na základě rozdílu hodnot pro službu od poslední aktualizace.

Takto vypočítáme rychlosti pro všechny stanice v obou směrech přenosu dat. Výslednou rychlost pro danou stanici v jednom směru bereme jako minimum ze všech služeb. Pokud je nová rychlost menší než minimální garantovaná rychlost, nastavíme maximum právě na garantované minimum. Tyto parametry jsou uloženy zpět do souboru s tabulkou o stanicích.

Nyní již známe všechny parametry, které jsou potřeba k nastavení jednotlivých tříd pro stanice v HTB. Nastavení rychlosti provádíme na obou rozhraních směrovače viz Obr. 1 - Model testovací sítě. Síťové rozhraní eth0 nám omezuje rychlosti pro upload, eth1 naopak pro download.

Na „Obr. 15 - FUP - Omezení šířky pásma klientské stanice“ je zobrazeno automatické omezování šířky pásma klientské stanice ve vnitřní síti na základě přenesených dat. Zelená hraniční křivka vymezuje maximální rychlost pro klientskou stanici pro ingress, modrá křivka pro egress. Porovnáním křivek v nočních a denních hodinách je pozorovatelný vliv koeficientu jednotlivých časových zón. Zároveň je patrné odlišné nastavení datových limitů pro oba směry. Na modré křivce pak dochází ke stagnaci na minimální hodnotě (bráno v absolutních hodnotách), která odpovídá minimální garantované rychlosti pro danou stanici.

V čase 00:01 dochází k obnovení nastavení rychlostí implicitními hodnotami, čehož je dosaženo pouze přepsáním souboru s tabulkou uživatelů a opětovným nastavením shaperu.



Obr. 15 - FUP - Omezení šířky pásma klientské stanice

Kompletní nastavení QoS je pak nastíněno podrobněji v příloze A nebo přímo viditelné ze skriptu Wspclient na přiloženém CD. Kompletní možnosti HTB je pak možno získat na webových stránkách autora [9].

6 Porovnání výkonnosti

K porovnání výkonnosti původního řešení realizovaného v rámci ročníkového projektu a nově vytvořeného v diplomové práci jsem na stejném počítači[2] provedl zvlášť operace čtení a nastavování struktur Iptables.

V obou případech se provádělo kompletní nastavení struktury Iptables pro 13 stanic v lokální síti uložené v souboru s informacemi o stanicích. Pro každou stanici byla vytvořena struktura řetězců a pravidel tak, aby bylo možné rozlišovat datové přenosy aplikačních protokolů. V projektu se rozlišovalo 5 skupin aplikačních protokolů, v diplomové práci jich rozlišujeme 9.

Projekt:

Ke čtení informací z Iptables byl použit původní skript „generator.pl“ s parametrem *„getdata“*. Sekce pro vytváření grafů nebyla prováděna, aby porovnání bylo věrohodné. Dochází ke čtení 2*5*13 = 130 hodnot počítadel.

Celkový čas čtení dat pro všechny stanice je 53s.

Nastavení struktur Iptables se provedlo spuštěním skriptu s parametrem „setfirewall“.

Nastavení struktur Iptables trvalo 49s.

Diplomová práce:

V diplomové práci byla struktura Iptables rozšířena o rozpoznání více skupin aplikačních protokolů (celkově 9 z původních 5). Pro 13 stanic tedy dochází ke čtení 2*9*13 = 234 hodnot počítadel. Pomocí klientské aplikace dojde k autorizaci spojení, pomocí cyklu postupné dotázání na hodnoty počítadel pro všechny stanice a odpojení.

Celkový čas čtení dat pro všechny stanice je 2,80s.

Nastavení struktur se provádí spuštěním aplikace Wsdemon s parametrem *„-SC“*

Nastavení struktur trvalo 1,7s.

2 Počítač používaný jako směrovač. Procesor Intel Pentium Celeron s frekvencí 300 MHz, 96 MB SDRAM.

7 Konfigurace

7.1 Konfigurace operačního systému

Zde je uveden přehled použitých technologií a balíčků, které je potřeba nainstalovat do operačního systému pro zajištění funkčnosti jednotlivých částí diplomové práce. Zásadní záležitostí je správně nakonfigurované jádro systému (nejméně řady 2.4, velmi však doporučuji řadu 2.6). Příloha B obsahuje konfiguraci jádra, kterou je nutno aktivovat.

7.1.1 Nutné součásti systému

Pro vytvořené aplikace je nutné mít v systému nainstalované tyto součásti:

Wsdemon

Iptables, L7filter, L7protocols

Wspclient

RRDTool, MD5 Perl Module

Klient pro vizualizaci Realtime statistik (realtimeajax.php, CSModul.php, kresligraf.php)

Web server (Apache), PHP (s podporou session, sockets, gd, cli, ctype), PEAR-Net_Sockets, PEAR-Image_Graph

Webový prohlížeč podporující technologii AJAX.

7.1.2 Konfigurace systému

Automatické vytváření grafů dlouhodobých statistik provedeme úpravou konfiguračního souboru pro "cron" démona přidáním řádku:

*0-59/2 * * * * root /data_app/wstat/wspclient.pl csg >/dev/null 2>&1*

To nám zajistí každé 2 minuty spuštění skriptu Wspclient s parametrem "csg" pro generování grafů do adresáře definovaného ve skriptu. Výstup posíláme na „/dev/null", aby nedocházelo k vypisování informací na konzoli.

Aktivaci FUP systému pak provedeme přidáním řádku:

*2 1-23/2 * * * root /data_app/wstat/2h_correctspeed.sh >/dev/null 2>&1*

Skript 2h_correctspeed.sh obsahuje dva řádky:

/data_app/wstat/wsdemon -NS 500 200

/data_app/wstat/wspclient.pl shaper

Každé 2h se provede aplikací Wsdemon přepočítání maximálních hodnot pro všechny stanice v síti (v ukázce s datovým limitem 500 MB pro download a 200 MB upload). Ve skriptu wspclient.pl s parametrem *"shaper"* jsou pak tyto datové limity aplikovány do jádra systému.

7.2 Konfigurace aplikace Wsdemon

Pro konfigurování parametrů aplikace Wsdemon slouží soubor „wstat.config", který je nutné mít umístěný v adresáři „/etc/wstat". Při spuštění aplikace dochází ke čtení jednotlivých parametrů z tohoto souboru.

Syntaxe pro nastavení je *název_proměnné="hodnota proměnné";*

Vysvětlení jednotlivých parametrů ukázkou konfigurace:

- NETWORK_CLIENTS_FILE="/etc/wstat/clients.lst";
  Cesta k souboru obsahující tabulku s informacemi o klientských stanicích.

- PASS_FILE="/etc/wstat/.auth";
  Cesta k souboru obsahující zašifrované heslo sloužící pro autorizaci klientských aplikací.

- NETWORK_CLIENTS_COUNTERS_FILE="/etc/wstat/clients.counters";
  Cesta k souboru, který je generován aplikací za účelem uložení hodnot statistik.

- EXTIF="eth0";
  EXTIP="86.118.90.11";
  INTIF="eth1";
  INTIP="10.1.1.1";

  Definice IP adres a označení síťových karet směrovače. EXTIF je síťové rozhraní ležící na straně Internetu a EXTIP jeho IP adresa. INTIF a INTIP jsou naopak údaje o síťovém rozhraní do vnitřní sítě.

- SERVERPORT="7896";

  TCP port na, kterém server aplikace naslouchá pro příchozí požadavky.

Následují konfigurační údaje slouží pro FUP algoritmus:

- TIMEZONES="8,16,23";
  KT="0.8,1,0.2";

  Definuje časové zóny a koeficienty pro každou z nich. V tomto případě odpovídá časové zóně 8-16h koeficient 0.8, 16-23h koeficient 1 a zóně 23-8h koeficient 0.2. Koeficient pak vyjadřuje jak hodně chceme v danou dobu omezovat rychlost.

- L0="1" koeficient skupiny služeb L0
  L1="0.7" koeficient skupiny služeb L1
  L2="0.1" koeficient skupiny služeb L2

- KS="L0,L1,L2,L0,L1,L0,L1,L1,L1";

  KS přiřazuje jednotlivé služby
  "HTTP","FTP","P2P","MAIL","SSL","REMOTE","AUDIO","VIDEO","OT"
  daným skupinám L0,L1,L2

- DATALIMIT_DOWN="200";
  DATALIMIT_UP="100";
  Definice datových limitů v megabytech pro oba směry.

8 Závěr

Díky předchozí znalosti o dané problematice se podařilo splnit všechny zadané cíle diplomové práce. Základ celého systému byl změněn tak, aby bylo možné monitorovat datové přenosy rozsáhlejších sítí. V rámci menší lokální sítě bylo monitorování odzkoušeno i s nasazením QoS systému s podporou FUP, které poskytuje konfigurovatelné automatizované řešení kritických situací v síti.

Při řešení aplikace Wsdemon a vizualizačních skriptů pro realtime statistiky jsem se potýkal s problémy chybějící dokumentace. Knihovny libiptc jsou experimentální a neexistuje pro ně žádná dokumentace. Rozšiřující modul PEAR-Image_Graph pro PHP použitý pro generování grafů je stále ve stádiu vývoje a dokumentace existuje jen ve velmi omezeném rozsahu. Často bylo tedy nutno nastudovat dané nástroje pomocí zdrojových kódů.

Využitím architektury klient server je možné zdokonalovat jednotlivé části systému zcela nezávisle. Dodržením použité syntaxe komunikace při síťové komunikaci klient-server lze systém rozšířit o další funkce a možnosti.

Do budoucna by bylo vhodné rozšířit řešení o možnost administrace systému pomocí webové aplikace, kde by mohl uživatel zadávat informace o stanicích sítě, konfigurovat parametry QoS systému a upravovat firewall směrovače. K ukládání informací o stanicích v síti do souboru využít databázový systém.

9 Použitá Literatura

[1] W. R. Stevens: TCP/IP Illustrated, Addison-Wesley, 1993

[2] Ch. E. Spurgeon: Ethernet, the definitive guide, O'Reilly, 2000

[3] Netfilter.org Project (www.netfilter.org)

[4] B.W Kernighan, D. M. Ritchie: Programovací jazyk C, Grada, 2006

[5] L7-filter Project (l7-filter.sourceforge.net)

[6] Internet Assigned Numbers Authority (http://www.iana.org)

[7] TCP/IP Sockets in C: Practical Guide for Programmers, Michael J. Donahoo, 2004

[8] RRDTool Project (http://oss.oetiker.ch/rrdtool/)

[9] HTB Home (http://luxik.cdi.cz/~devik/qos/htb/)

[10] Průchod paketů strukturou Iptables (http://www.linuxhomenetworking.com)

10 Obsah přiloženého CD

Obsah kořenového adresáře přiloženého CD:

/etc/wstat	Ukázkové konfigurační soubory
/Dokumentace	Text diplomové práce
/RealtimeGraphsClient	Klientské aplikace
/Wsdemon	Aplikace Wsdemon
/WspClient	Klientská aplikace

Přílohy

A) Nastavení HTB

Ukázkové nastavení pro jedno síťové rozhraní:

Na kořenový uzel (root node) jsem zapojil třídu pojmenovanou 1:1 s maximální propustností 100 Mbit/s. Z ní se pak větví další 2 třídy 1:10 a 1:20, každá s garantovanou propustností 256 kbit a maximální rychlostí 100 Mbit. Jedna je určena pro provoz po vnitřní síti a druhá pro provoz do internetu.

Na kořen zapojíme filtr, který nám tyto 2 druhy provozu oddělí a odešle do správné třídy. Na třídu určenou od internetu (1:10) přidáme další kořen, který bude mít maximální rychlost 3980 kbit (pro případ 4 Mbit linky do internetu) a na tento pak pro každého uživatele vytvoříme jednoho potomka (10:11x), který bude mít garantovanou minimální rychlost (ta se počítá v závislosti na počtu uživatelů a maximální rychlosti linky), maximální rychlost a prioritu (získaná dotazem na server Wsdemon).

Maximální rychlost celého tohoto podstromu na 3980 kbit jsme nastavili z toho důvodu, aby se fronta paketů tvořila na naší straně sítě. To je nutná podmínka pro to, aby shaper fungoval správně.

Na tento podstrom připojíme ještě další dva potomky. První (10:11) je určen pro prioritní provoz DNS dotazů, synchronizačních paketů a SSH provoz. Do druhého (který má nastavenou velmi malou garantovanou i maximální rychlost a nízkou prioritu) spadají všechny pakety, které nevyhověly zadaným filtrům pro jednotlivé uživatele a pakety zakázaných protokolů jednotlivých uživatelů. Místo obyčejné FIFO fronty na výstupu přidáme na koncové listy tohoto stromu SFQ fronty, které jsou pak cyklicky obsluhovány tak, že z každé fronty je vybráno stejné množství dat.

Jelikož je možné těmito frontami omezovat rychlost pouze na výstupu, vytvoříme tuto strukturu pro obě rozhraní. Síťové karta na straně internetu (eth0) nám omezuje rychlost do internetu - UPLOAD a síťová karta na lokální straně omezuje příchozí rychlost paketů z internetu - DOWNLOAD.

Při použití NAT vzniká problém s filtrováním paketů pomocí U32 filtrů (viz U32 filtr níže), protože hlavičky paketů prošlé NAT tabulkou už mají změněné zdrojové adresy (viz Obr. 3 - Průchod paketů strukturou Iptables [10]).

U32 filtr:

`$TC filter add dev eth1 parent 10: prio 2 protocol ip u32 match ip dst $_[1]"/32" flowid 10:11$actualuser`;

Proto je nutné pro filtrování paketů ve směru do internetu každý paket označkovat příslušným číslem (podle zdrojové adresy) v PREROUTING řetězci mangle tabulky:

`$IPTABLES -t mangle -A PREROUTING -s $_[1] -d ! 10.1.1.0/24 -j MARK --set-mark $actualuser`;

Tyto pakety pak právě pomocí značek můžeme řadit do správných tříd:

`$TC filter add dev eth0 parent 10: prio 2 protocol ip handle $actualuser fw flowid 10:11$actualuser`;

Přiřazení tříd použitím modulu CLASSIFY:

`$IPTABLES -t mangle -A PREROUTING -s $_[1] -d ! 10.1.1.0/24 -j CLASSIFY --set-class 10:11$actualuser`;

Kompletní nastavení QoS pro řízení je vidět ve zdrojovém kódu skriptu „Wspclient.pl“ na přiloženém CD.

B) Konfigurace jádra

Network Options -> Network packet filtering

-> Core Netfilter configuration->

- Netfilter Xtables support
- CLASSIFY target support
- CONMARK target support
- „conmark“ connection mark match support
- MARK target support
- MAC match support

-> IP: Netfilter Configuration

- Connection tracking
- Connection mark tracking support
- IP tables support
- TOS match support
- Packet filtering
- Full NAT
- Packet mangling
- TOS target support

-> QoS and/or fair Queueing->

- HTB
- SFQ
- Netfilter Mark
- U32 key

Cryptographic Options -> MD5